**REAL-TIME HEADING ESTIMATION USING PERSPECTIVE FEATURES**

THESIS

James W. Dean, Captain, USAF

AFIT-ENG-13-M-13

AFIT-ENG-13-M-13

REAL-TIME HEADING ESTIMATION USING PERSPECTIVE FEATURES

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Insitute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

James W. Dean, B.S.E.E.

Captain, USAF

March 2013

AFIT-ENG-13-M-13

REAL-TIME HEADING ESTIMATION USING PERSPECTIVE FEATURES
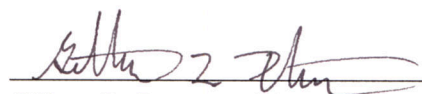
James W. Dean, B.S.E.E.
Captain, USAF

Approved:

_____       7 MAR 2013

John F. Raquet, PhD (Chairman)         Date

_____       7 MAR 2013

Gilbert L. Peterson, PhD (Member)      Date

_____       7 Mar 2013

Lt Col Kenneth A. Fisher, PhD (Member)    Date

# Abstract

There are a large number of commercially available quad-rotor helicopters available from various manufacturers. All of these systems rely on a low cost MEMS based inertial measurement system for stabilization and navigation. These low cost inertial systems are all subject to rapid error growth in their attitude and position estimates unless bounded by external measurements. This thesis created real-time algorithm to integrate measurements from visual cues with measurements from onboard sensors to estimate the attitude position and velocity of a quad-rotor helicopter in a local navigation frame, a system model for the ARDrone, and a feed-back controller for the vehicle's heading.

The ARDrone, by Parrot SA, is a low cost quad-rotor helicopter that comes equipped with a variety of sensors including a forward-looking high-definition camera. The vehicle is capable of using its onboard sensors to adequately constrain the errors for pitch and roll in all environments, however the yaw axis is still subject to drift. This work utilizes a RANSAC based vanishing point detection algorithm to provide a reliable heading reference and integrates the vanishing point based heading measurements with the system's on-board heading measurements through an extended Kalman filter. In addition to estimating the drone's heading, the Kalman filter also estimates the position and velocity of the drone as it moves through its environment.

This system was able to provide a heading reference with an error of one degree for the drone and was shown to be capable of transitioning between vanishing points when the vehicle needed to change direction. The system also demonstrated that it was capable of generating an estimate of position and velocity. However because the position error was on the order of one meter, the estimate was not accurate enough for autonomous navigation. With an additional source of velocity measurements the vehicle could be a capable, low cost solution for mapping man-made structures.

# Table of Contents

# List of Figures

# List of Tables

# List of Symbols

| Symbol | Definition |
|--------|------------|
| $\phi$ | Roll (rotation about the X axis) |
| $\theta$ | Pitch (rotation about the Y axis) |
| $\psi$ | Yaw (rotation about the Z axis) |
| $\delta\psi$ | Heading Error |
| $\Psi_R$ | Heading reported by the onboard INS |
| $\Psi_V$ | Heading based on a vanishing point |

*Subscripts*

| | |
|--------|------------|
| *MEAS* | Measurement |
| *X* | X Component |
| *Y* | Y Component |

*Superscripts*

| | |
|--------|------------|
| *B* | Body Reference Frame |
| *N* | Navigation Reference Frame |
| *C* | Camera Reference Frame |
| *PIX* | Pixel Reference Frame |

# List of Acronyms

| Acronym | Definition |
|---------|------------|
| INS | Inertial Navigation System |
| SDK | Software Developers Kit |
| EKF | Extended Kalman Filter |
| MEMS | Micro-Electro-Mechanical Systems |
| RANSAC | Random Sample Consensus |
| GUI | graphical user interface |
| GPS | Global Positioning System |

REAL-TIME HEADING ESTIMATION USING PERSPECTIVE FEATURES

# 1   Introduction

T HE autonomous operation of small aerial vehicles inside man-made and natural structures has many challenges. Many of the traditional navigation aids such as the Global Positioning System (GPS) are not available. Relying on inertial measurements alone is also not feasible, because errors in the inertial measurements cause the system navigation errors to grow unbounded [22]. Combining measurements from a video camera with the inertial measurement system can yield an effective navigation system. The goal of this thesis is to develop a real-time algorithm that combines information from a forward looking camera with onboard measurements to estimate the attitude, position and velocity of a vehicle.

## 1.1   Background

In recent years, a number of low-cost quad-rotor helicopters (quad-copters) have become available on the commercial market. Companies such as MikroKopter [2] and Ascending Technologies [1] both offer multi-rotor helicopters designed for research and development. These systems can be costly and are easily damaged. They also require safety nets to be flown indoors, which makes them poor candidates for indoor exploration and mapping missions. Parrot SA developed the ARDrone to be flown indoors.

The ARDrone is a small quad-copter that is designed to be flown indoors and controlled by a smart phone or tablet computer. The ARDrone is equipped with a set of inertial sensors, altitude sensors, optical-flow sensor, and a high definition video camera.

Combined with its open source ground control software the ARDrone makes an attractive platform for conducting indoor navigation research.

## 1.2 Problem

One problem facing small airborne vehicles operating inside a man-made structures is localization. Many of the techniques developed for wheeled robots are often impractical or not feasible for implementation on an airborne platform. Wheeled robots, for example, can count wheel revolutions to obtain a reasonably accurate estimate of how far they have traveled. Using a simple touch sensor, a wheeled robot can build a map of its environment by simply tracking the distance between objects it collides with. For airborne vehicles that are not in contact with the ground, measuring distance becomes much more complicated, and bumping into obstacles is often enough to cause the vehicles to crash. A common solution to these problems is to equip air vehicles, such as quad-copters, with active sensors such as laser or ultrasonic range finders. However, these sensors add weight and power requirements that reduce the vehicle's maneuverability and flight time. The ideal solution would utilize low-power passive sensors, such as a video camera, that are often already present on the vehicles.

The problem this research seeks to address is using the video camera already present on a vehicle to aid its on-board navigation system in order to improve its ability to localize itself, and map an unknown environment. In order to do this, vanishing points will be used to provide an accurate heading estimate which will enable more accurate mapping using the vehicle's velocity measurement system.

## 1.3 Scope

Vision based navigation for mobile robots in an indoor environment is a broad and very active area of research. The techniques developed to date fit into three categories:

map-based navigation, map-building-based navigation, and mapless navigation [8]. This topic area has been broadly surveyed by the authors of [8] and [4].

In map-building-based navigation, the robot builds a geometric model as it observes features in its environment. These techniques work well when the robot has a reliable set of sensors that it can use to observe the environment. A limitation of map-building techniques is that any uncertainty in the measurement system that is not accounted for is compounded over time. Examples of systems using map-building techniques can be found in [6], [21], and [23].

A robot using mapless navigation techniques makes navigation decisions based on what the robot is currently observing. These algorithms work well when the robot is simply required to move through an environment and avoid obstacles. Because the robot does not need to store any map information, the computational resources required for some of the algorithms are lower than the other two techniques. A limitation of mapless navigation is that the vehicle is always operating in an exploration mode. It can't apply any of its past observations to future route planning. As a result, the robot can spend a large amount of time wandering around a relatively small area. Examples of systems that use mapless navigation techniques can be found in [11], [16], and [19].

This research is focused on combining the on-board attitude and velocity measurements of the ARDrone with information extracted from a forward facing video camera to estimate the vehicle's trajectory through a hallway. It will not address map-building or decision making specifically.

## 1.4 Related Work

A problem common to all inertial navigation systems, regardless of size and cost, is that they are all subject to drift over time. The notion of using visual cues from an on-board camera to aid the on-board inertial sensors is not a new concept. This research follows on research done by Prahl in [18].

Prahl focused on demonstrating that vanishing points extracted from a perspective image could be used to constrain the errors in a Micro-Electro-Mechanical Systems (MEMS) grade Inertial Navigation System (INS). This was done by extracting three vanishing points from a perspective image. Because the environment was assumed to meet the assumptions of the Manhattan World model, three orthogonal vanishing points could be extracted from each image. In the Manhattan world model, it is assumed the floor and ceiling meet the walls at a 90°, and that all the corridors in the building intersect at a 90°. The vectors pointing to the vanishing points were used to form an earth fixed local level navigation frame. This frame was then used to estimate the systems attitude. An extended Kalman filter was then used to combine the attitude estimate from the vanishing point algorithm with the attitude estimate from the MEMS based INS. Prahl's experiment consisted of an ADIS 16355 MEMS based INS and a small USB video camera mounted to a cart. A laptop was used to record video and INS measurements as the cart was pushed around a test course. A Honeywell HG1700-AG58 tactical grade INS was also attached to the cart to provide truth data. Prahl's algoirthm was run in a post-processing mode on the recorded data. The results of Prahl's experiments showed that his algorithm could generate an attitude estimate with with an average error of $\pm 1.5°$ when compared to the tactical grade INS.

Prahl's work demonstrated the feasibility of using vanishing points to constrain the errors inherent in a navigation system based on inertial sensors. In [3], vanishing point estimation was used to guide an ARDrone through a corridor in a building. Bills [3] developed a real-time algorithm that located the vanishing point at the end of a corridor and used proportional control to keep the vanishing point centered in the ARDrone's forward-looking camera as it flew.

## 1.5   Research Contributions

This research will contribute a system model for the ARDrone, a computer vision algorithm for measuring heading in real-time, a state estimation algorithm based on the previously developed system model. All of the algorithms developed in this research run in real-time and directly control the vehicle as it flies through the hallway.

## 1.6   Research Overview

The research conducted begins with creating a software design to support the test and evaluation of the vision aided navigation algorithms. Once the necessary software design is in place, the vanishing point detection algorithm developed in [18] is adapted to provide realtime heading measurements. A Kalman filter is designed to integrate the ARDrones navigation measurements and the computer vision measurements to estimate the drone's attitude, position and velocity. After tuning the system for performance, a series of flight tests are conducted to validate the system design and characterize its performance. The research concludes with some additional tuning is conducted by post-processing the measurements recorded during the flight tests.

## 1.7   Thesis Overview

Chapter 2 includes background information to aid in understanding the algorithms developed for this research. In Chapter 3, the system model of the ARDrone and the algorithms developed for this research are presented. Chapter 4 contains a description of the experiments developed to evaluate the performance of the algorithms developed in Chapter 3 and the results. The final chapter covers the conclusions reached by the author and suggestions for future research.

## 2 Background

THIS chapter presents a brief review of several concepts that are used frequently in this document. The first section reviews the reference frames used in this text, as well as some common transformations used to convert between reference frames. Section 2 covers topics related to computer vision topics; including an overview of vanishing points, camera calibration, and image rectification. The conclusion of this chapter is a review of Kalman filtering and residual monitoring. A table of notations and conventions used can be found in Table 2.1

### 2.1 Reference Frames

Measurements of position and orientation are taken relative to the reference frame for the measurement device. When measurements from multiple devices need to be combined, they must be combined in a common frame of reference. This section contains a brief overview of coordinate transforms as presented in [9].

A reference frame is defined by an origin and a number of orthonormal vectors known as basis vectors. The number of basis vectors depends on the dimensionality of the reference frame. For example, a two-dimensional plane requires two basis vectors, three dimensions require three basis vectors, etc. The coordinates of any point in the reference frame can then be expressed as the lengths of the orthogonal projections of the vector between that point and the origin on to each of the basis vectors. The same point can have different representations based on the reference frame used to describe the location of the point.

Converting between two reference frames is done by a rigid transformation. The rigid transform for a point $P$ in frame A is expressed as

$$P^A = \mathbf{C}_B^A P^B + \mathbf{t}^A \tag{2.1}$$

Table 2.1: The notation and conventions used in this document.

| Notation | Description |
| --- | --- |
| $\mathbf{a}$ | Vectors are denoted using bold face lower case letters. |
| $P$ | Points are denoted using capitol letters. |
| $\mathbf{a} = \{a_x, a_y, a_z\}$ | Vector components are denoted using the same lower case letter as the vector, with lower case subscripts identifying the component. |
| $\mathbf{A}$ | Matrices are denoted using capital, bold face letters. |
| $\mathbf{a}^B$ | Capital superscripts are use to denote the reference frame used to express the vector. |
| $P^{PIX}$ | A super script $PIX$ is used to designate the pixel frame. |
| $P^C$ | A superscript $C$ is used to designate the camera frame. |
| $P^B$ | A superscript $B$ is used to designate the body frame. |
| $P^N$ | A superscript $N$ is used to designate the navigation frame. |
| $\mathbf{C}_B^A$ | Rotation matrices represent a rotation from the subscripted coordinate frame to the superscripted coordinate frame. |
| $\hat{\mathbf{a}}$ | Estimated values of quantities are denoted with the hat character. |
| $\hat{f}(t^-)$ | A priori estimates of the value of a function or variable are denoted using the "minus" superscript. |
| $\hat{f}(t^+)$ | A posteriori estimates of the value of a function or variable are denoted using a superscript "plus". |
| $\underline{\mathbf{s}}$ | Vectors expressed in homogenous coordinates are denoted with an underline accent. |

where $P^A$ is the point's coordinates in the $A$ frame of reference, $\mathbf{C}_B^A$ is the rotation matrix, and $P^B$ is the point's coordinates in the $B$ frame of reference. The vector $\mathbf{t}^A$ is the translation

vector expressed in the *A* frame of reference. The rotation matrix is defined as

$$C_B^A = \begin{bmatrix} \mathbf{i}^A & \mathbf{j}^A & \mathbf{k}^A \end{bmatrix} \tag{2.2}$$

where the vectors $\mathbf{i}^A$, $\mathbf{j}^A$, and $\mathbf{k}^A$ are the basis vectors of the *B* frame expressed in the *A* frame. When the reference frames are comprised of orthonormal vectors in a right handed orientation a rotation matrix as Two useful properties; its inverse is equal to its transpose, and its determinant is equal to one [9]. The rotation and translation operations are summarized in Figure 2.1.
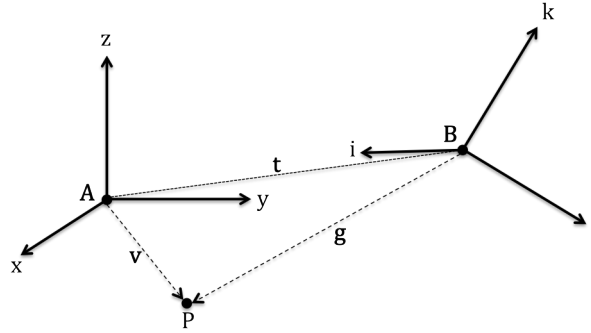


Figure 2.1: The rigid transform operation. A point *P* can be described in terms of either the *A* frame of reference or the *B* frame of reference.

### 2.1.1 Common Reference Frames.

**The Pixel Frame.** This frame is the coordinate frame for images captured from a camera. Pictured in Figure 2.2, the pixel frame is a two-dimensional frame where the positive *x*-axis represents the columns of pixels and the positive *y*-axis represents the rows of pixels. The origin of this coordinate frame is traditionally placed in the upper left-hand corner, although some applications place it in the lower left-hand corner.

**The Camera Frame.** This three-dimensional frame is shown in Figure 2.3. In the camera frame, the positive *z*-axis extends out through the center of the lens, the positive

8
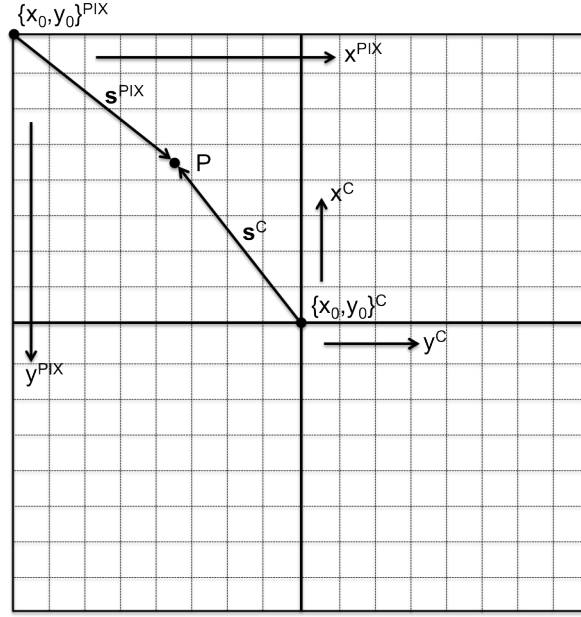
Figure 2.2: The pixel coordinate frame is the coordinate frame for images captured by a digital camera. For a point, P, the vector $\mathbf{s}^{PIX}$ is equivalent to $T_C^{PIX}\mathbf{s}^C$.

$x$-axis extends vertically and the positive $y$-axis extends to the right. The origin is fixed at the camera's focal point.

***The Body Frame.*** This reference frame, pictured in Figure 2.4, is important because measurements provided by the drone are expressed in this reference frame. In addition to the axes, the three Euler angles, roll, pitch and yaw, are defined in this coordinate frame. Roll represents rotation about the $x$-axis, pitch represents rotation about the $y$-axis and yaw represents rotation about the $z$-axis.

***The Navigation Frame.*** This frame is an earth fixed, local level reference frame. The navigation frame's origin and orientation are determined when the drone is initialized. The navigation frame's origin is fixed at the point where the drone is initialized, and the navigation frame's $x$-axis is parallel with the $x$-axis in the drone's body frame. As the
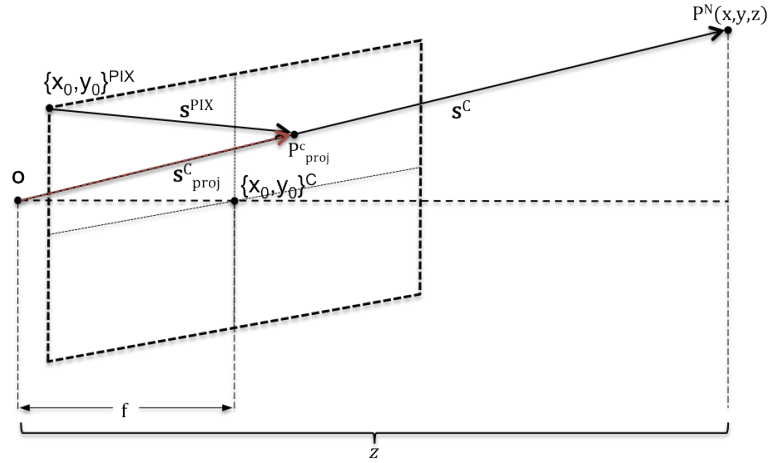
Figure 2.3: This image shows the relationship between the camera frame and the pixel frame. Because the pixel frame is a two-dimensional frame, points in the pixel frame are projected onto a virtual image plane one focal length in front of the origin in the camera frame.
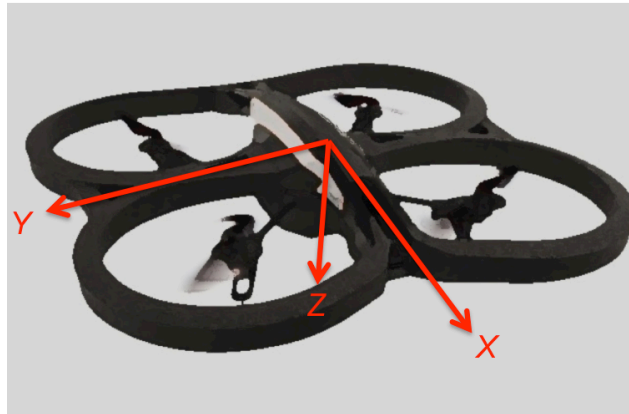


Figure 2.4: In the body frame, the $x$-axis extends through the nose of the aircraft, the y-axis extends out of the left wing, and the $z$-axis extends from the belly of the aircraft.

aircraft moves through the environment, the navigation frame remains fixed and is used to measure the aircraft's movement relative to its initial position.
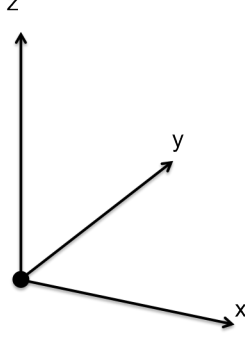
Figure 2.5: The navigation frame is a right-handed coordinate system. The origin of this frame is fixed at the point where the aircraft is initialized

### 2.1.2 Coordinate Frame Transformations.

Coordinates in the body frame are transformed to and from the navigation frame using the rotation matrix

$$C_B^N =$$

$$\begin{bmatrix} \cos(\theta)\cos(\psi) & -\cos(\phi)\sin(\psi) + \sin(\phi)\sin(\theta)\cos(\psi) & \sin(\phi)\sin(\psi) + \cos(\phi)\sin(\theta)\cos(\psi) \\ \cos(\theta)\sin(\psi) & \cos(\theta)\cos(\psi) + \sin(\phi)\sin(\theta)\sin(\psi) & -\sin(\phi)\cos(\psi) + \cos(\phi)\sin(\theta)\sin(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix}$$

$$(2.3)$$

where $\phi$ represents roll, $\theta$ represents pitch, and $\psi$ represents yaw [22].

Coordinates in the camera frame are converted to and from the body frame using the rotation matrix

$$C_C^B = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad (2.4)$$

where $C_C^B$ is the camera to body rotation. Because the feature of interest lies at infinity, the translation vector between the drone's frame origin and the camera's focal point is neglected.

11

The coordinates in the pixel frame are transformed to and from the camera frame using the camera matrix

$$\mathbf{T}_C^{PIX} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

where $f_x$ and $f_y$ are the focal lengths in the $x$ and $y$ directions. The skew coefficient, represented by $\gamma$, accounts for cases where $f_x \neq f_y$. The terms $c_x$ and $c_y$ are coordinates of the camera's principal point. These parameters are generated by the camera calibration process described in Section 2.2.2.

## 2.2 Computer Vision

Computer vision is a broad category of algorithms that all have as their goal extracting information about the environment from digitized images. This section contains a brief overview of the camera model, algorithms used for detecting edges and extracting lines from a digital image. These algorithms are then applied to the problem of vanishing point detection.

### 2.2.1 Vanishing Points.

Under perspective projection, lines that are parallel in nature appear to intersect at a common point on the horizon [10], as shown in Figure 2.6. In the Manhattan world, a set of mutually orthogonal vanishing points can be located. A set of vectors pointing to these orthogonal vanishing points can then be used to construct a basis set for the navigation frame.

### 2.2.2 Camera Calibration.

The camera calibration process is used to determine the intrinsic characteristics of the camera. The intrinsic characteristics are the camera's focal length, principal point, and distortion coefficients. A common method for determining these parameters is through an iterative optimization process described in [24].
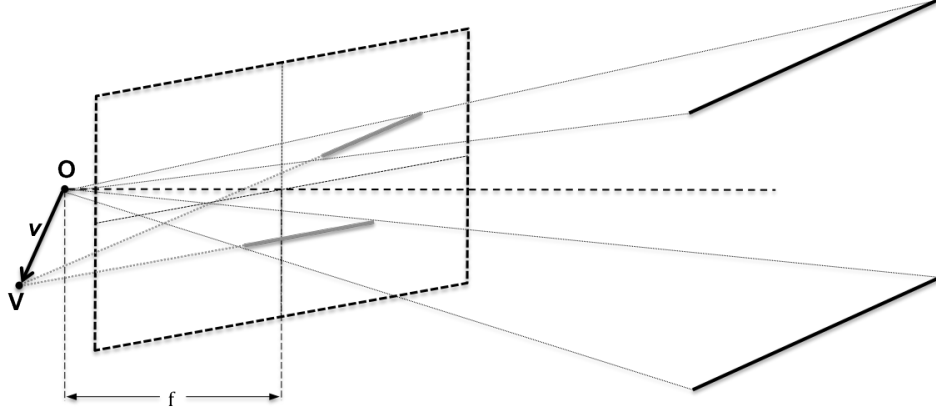
Figure 2.6: Under perspective projection two parallel lines in a scene appear to intersect at a point on the horizon, the vanishing point **V**. Vector *v* can be used to form a basis set for a reference frame.

The projection model used by Zhang [24] is

$$s\underline{\mathbf{m}} = \mathbf{A}\,[\mathbf{R} \quad \mathbf{t}]\,\underline{\mathbf{M}} \tag{2.6}$$

where *s* is a arbitrary scale factor, $\underline{\mathbf{m}}$ represents the homogenous pixel coordinates of an object, **A** is the camera's intrinsic parameters, [**R** **t**] are the extrinsic pararmeters, and $\underline{\mathbf{M}}$ is the homogenous coordinates of an object in the world frame. The intrinsic parameters represented by **A** are the same as the parameters in $\mathbf{T}_C^{PIX}$. The extrinsic parameters of the camera are comprised of **R**, the rotation between the camera and the object, and **t**, the translation between the camera and object.

The calibration process uses multiple images of a planar object with known feature dimensions to determine the camera's intrinsic, extrinsic, and distortion parameters using maximum likelihood estimation. Each image has a different view of the object. The motion of the camera between images does not need to be captured, but must be comprised of both rotation and translation. For this research the planar object used to calibrate the camera is

a square checkerboard. The feature points used are the interior corners of the checkerboard squares. Example images from a calibration set are shown in Figure 2.7.
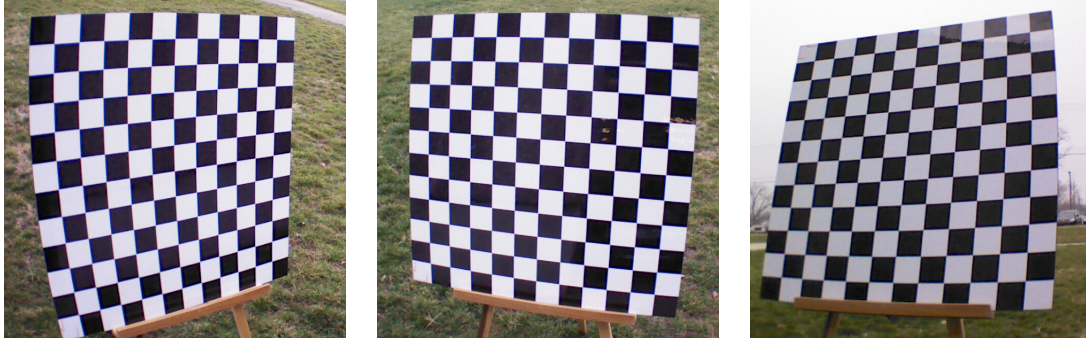


Figure 2.7: Example images from a calibration set. The checkerboard is comprised of a 12 x 12 array of squares that are 76mm x 76mm in size.

### 2.2.3 *Image Rectification.*

The camera projection model presented in this section assumes a linear mapping between points in the scene and pixels on the image plane. For most cameras this is not an accurate assumption, because of distortions placed on the image by the lens. Before the translations described in previous sections can be applied, the image must be rectified.

Image rectification applies a distortion model to correct the location of each pixel in the image. A radial distortion model is

$$\hat{x} = x_c + L(r)(x - x_c) \text{ and } \hat{y} = y_c + L(r)(y - y_c) \tag{2.7}$$

where $x$ and $y$ are the original (distorted) pixel coordinates, $\hat{x}$ and $\hat{y}$ are the corrected pixel coordinates, $L(r)$ is the distortion model. The $x_c$ and $y_c$ parameters represent the center of distortion [10]. The camera's principal point is used as the center of distortion in this work. The image distortion model used in this work is

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \kappa_4 r^4 + \kappa_5 r^5 \tag{2.8}$$

14

where $r = \sqrt{x^2 + y^2}$, and $\kappa_1$ through $\kappa_5$ are the distortion coefficients determined by the camera calibration process.

## 2.3 Computational Complexity

Computer algorithms are often classified in terms of their asymptotic order of growth. An algorithm's order is used to describe its execution time growth as a function of the size of the input. If an algorithm's execution time is described by a function $T(n)$ where $n$ is the size of the input, and there exists some $c$ that satisfies

$$T(n) \leq cf(n) \quad \text{where } c > 0 \tag{2.9}$$

then the algorithm is said to be $O(f(n))$[12].

## 2.4 Kalman Filtering

Deterministic system models are built from a system of linear, partial differential equations. In such models, each state can be precisely determined and is a function of the system dynamics and the inputs to the system. On systems such as quad-rotor helicopters it is not possible to precisely model the system dynamics. In addition to uncertainties in the system dynamic model, the measurements of the system states required for control are corrupted by noise and bias. In some systems it may be impractical or impossible to directly measure the desired states. The values of these states can be estimated based on states that can be measured. For example, there is often no way to measure absolute position in the desired frame of reference, but position can be inferred from knowledge of the vehicle's velocity. Another common problem arises when multiple measurements are available for the same state. Rather than ignoring redundant measurements, it is desirable to optimally combine these measurements. A common solution to all of these problems is to utilize a Kalman filter for state estimation. The following sections provide a brief discussion of the Kalman filter based on Dr. Peter Maybeck's work in [14] and [15].

The Kalman filter is a recursive estimation algorithm based on Baysian statistics that provides an optimal estimate of the mean and covariance of a stochastic linear system based on all available information. A stochastic linear system has the form of

$$\dot{\mathbf{x}} = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{G}(t)\mathbf{w}(t) \tag{2.10}$$

where $\mathbf{x}(t)$ is a vector of state variables, $\mathbf{u}(t)$ is a vector of deterministic inputs, and the process noise $\mathbf{w}(t)$ is a vector of white Gaussian noise terms that satisfy

$$E\{\mathbf{w}(t)\} = 0 \tag{2.11}$$

$$E\left\{\mathbf{w}(t)\mathbf{w}^T(t')\right\} = \mathbf{Q}(t)\delta(t - t') \tag{2.12}$$

where $E$ is the expectation operator, and $\delta(t)$ is the Dirac delta function. The function $\mathbf{Q}(t)$ is positive semidefinite for all values of t and is piecewise continuous over the interval $[t, t']$. The semidefinite nature of $\mathbf{Q}(t)$ allows for cases where states are known precisely.

For this work, the discrete time implementation of the Kalman filter algorithm is used. A description of the continuous time implementation can be found in[14]. The discrete time Kalman filter algorithm is comprised of two steps. First, the state statistics are propagated from the previous time step, then the estimates are corrected by combining them with the measurement data available at the current time step. If no measurements are available, propagating the states provides the best estimate of their current values. The states are propagated by

$$\hat{\mathbf{x}}(t_i^-) = \mathbf{\Phi}(t_i, t_{i-1})\hat{\mathbf{x}}(t_{i-1}^+) + \mathbf{B_d}(t_{i-1})\mathbf{u}(t_{i-1}) \tag{2.13}$$

where $\mathbf{\Phi}(t_i, t_{i-1})$ represents the discrete state transition matrix, and $\mathbf{B}_d$ is the discrete input matrix. The discrete state transition matrix describes how the states propagate over the time interval $[t_{i-1}, t_i]$, and is calculated using the continuous-time state transition matrix using

$$\mathbf{\Phi}(t_i, t_{i-1}) = e^{\mathbf{F}(t_i)(t_i - t_{i-1})} \tag{2.14}$$

where e is the Euler exponential. When $t_i - t_{i-1}$ is small relative to the terms in $\mathbf{F}(t)$ a first order approximation can be used to calculate $\mathbf{\Phi}(t_i, t_{i-1})$. The first order approximation

$$\mathbf{\Phi}(t_i, t_{i-1}) = \mathbf{I} + \mathbf{F}(t_i)\,(t_i - t_{i-1}) \tag{2.15}$$

where $\mathbf{I}$ is the identity matrix, is derived by neglecting the higher order terms of the Taylor series expansion of the matrix exponential function. The discrete input matrix is determined by

$$\mathbf{B_d}(t_i, t_{i-1}) = \int_{t_{i-1}}^{t_i} \mathbf{\Phi}(t, \tau)B(\tau)d\tau \tag{2.16}$$

where $\mathbf{B}(t)$ is the continuous-time input matrix. Here again, the first order approximation

$$\mathbf{B_d}(t_i, t_{i-1}) = \mathbf{B}(t_i)\,(t_i - t_{i-1}) \tag{2.17}$$

is often sufficient.

After propagating the state estimates, the state covariance is propagated by

$$\mathbf{P}(t_i^-) = \mathbf{\Phi}(t_i, t_{i-1})\mathbf{P}(t_{i-1}^+)\mathbf{\Phi}^T(t_i, t_{i-1}) + \mathbf{Q}_d \tag{2.18}$$

where $\mathbf{Q}_d$ is the discrete process noise covariance matrix. The discrete process noise covariance matrix is calculated by

$$\mathbf{Q_d}(t_i, t_{i-1}) = \int_{t_{i-1}}^{t_i} \mathbf{\Phi}(t, \tau)\mathbf{G}(\tau)\mathbf{Q}(\tau)\mathbf{G^T}(\tau)\mathbf{\Phi^T}(t, \tau)d\tau \tag{2.19}$$

The first order approximation

$$\mathbf{Q_d}(t_i, t_{i-1}) = \mathbf{G}(t_i)\mathbf{Q}(t_i)\mathbf{G^T}(t_i)\,(t_i - t_{i-1}) \tag{2.20}$$

can often be used.

After propagating the states and state covariance, the measurement data is used to correct the propagated states and update the state covariance. The measurement model takes the form of

$$\mathbf{z}(t_i) = \mathbf{H}(t_i)\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{2.21}$$

where $\mathbf{z}(t_i)$ is a vector of measurements, $\mathbf{H}(t_i)$ is the measurement matrix, and $\mathbf{v}(t_i)$ is a vector of zero-mean, white, Gaussian processes. The measurement matrix relates the measurements in $\mathbf{z}(t_i)$ to the states. The covariance of the measurement noise is

$$\mathbf{E}\left\{\mathbf{v}(t_i)\mathbf{v}^{\mathbf{T}}(t_j)\right\} = \begin{cases} \mathbf{R}(t_i) & t_i = t_j \\ \\ 0 & t_i \neq t_j \end{cases} \tag{2.22}$$

where $\mathbf{R}(t_i)$ is a symmetric positive definite matrix. The positive definite nature of $\mathbf{R}(t_i)$ prohibits precisely known measurements. One of the features of the Kalman filter is the ability to optimally combine all of the information provided by the measurements with the current state estimates. This is done through the Kalman gain matrix which is determined by

$$\mathbf{K}(t_i) = \mathbf{P}(t_i^-)\mathbf{H}^T(t_i)\left[\mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^T(t_i) + \mathbf{R}(t_i)\right]^{-1} \tag{2.23}$$

which preferentially weights terms with the lowest covariance. With the Kalman gain defined, the states can be corrected by

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)\left[\mathbf{z}_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)\right] \tag{2.24}$$

and the state covariance updated by

$$\mathbf{P}(t_i^+) = \mathbf{P}(t_i^-) - \mathbf{K}(t_i)\mathbf{H}(t_i)\mathbf{P}(t_i^-) \tag{2.25}$$

This process is repeated for each new measurement.

### 2.4.1   *The Extended Kalman Filter.*

The filter algorithm described in the previous section requires both the system dynamics model and the measurement model to be linear. However, not all systems can be adequately modeled by a linear system. The Kalman filter algorithm can be extended by linearizing the filter equations around a nominal trajectory. The system model becomes

$$\dot{\mathbf{x}} = \mathbf{f}\left[\mathbf{x}(t), \mathbf{u}(t), t\right] + \mathbf{G}(t)\mathbf{w}(t) \tag{2.26}$$

18

where the state transition matrix becomes a function of the states and the forcing function. The process noise terms remain the same as they were for the linear model. For the extended Kalman filter, the nominal trajectory is the last best estimate of the system state $\hat{\mathbf{x}}(t_i^-)$. The propagation equation becomes

$$\hat{\mathbf{x}}(t_{i+1}) = \hat{\mathbf{x}}(t_i^+) + \int_{t_i}^{t_{i+1}^-} \mathbf{f}\left[\hat{\mathbf{x}}(t_i^+), \mathbf{u}(t), t\right] dt \tag{2.27}$$

for the states and

$$\mathbf{P}(t_{i+1}^-) = \mathbf{\Phi}\left[t_{i+1}, t_i, \hat{\mathbf{x}}(\tau/t_i)\right] \mathbf{P}(t_i^+) \mathbf{\Phi}^{\mathbf{T}}\left[t_{i+1}, t_i, \hat{\mathbf{x}}(\tau/t_i)\right]$$

$$+ \int_{t_i}^{t_{i+1}} \mathbf{\Phi}\left[t_{i+1}, t_i, \hat{\mathbf{x}}(\tau/t_i)\right] \mathbf{G}(t)\mathbf{Q}(t)\mathbf{G}^{\mathbf{T}}(t)\mathbf{\Phi}^{\mathbf{T}}\left[t_{i+1}, t_i, \hat{\mathbf{x}}(\tau/t_i)\right] d\tau \tag{2.28}$$

where $\tau$ is the variable of integration, for the covariance. The linearized $\mathbf{\Phi}\left[t_{i+1}, t_i, \hat{\mathbf{x}}(\tau/t_i)\right]$ is determined by evaluating

$$\mathbf{F}\left[t; \hat{\mathbf{x}}(t_i^-)\right] = \frac{\delta \mathbf{f}\left[\mathbf{x}, \mathbf{u}(t), t\right]}{\delta \mathbf{x}}\bigg|_{\mathbf{x}=\hat{\mathbf{x}}(t_i^-)} \tag{2.29}$$

and substituting the resulting $\mathbf{F}\left[t, \hat{\mathbf{x}}(t_i^-)\right]$ into Equation (2.14).

The measurement model becomes

$$\mathbf{z}(t_i) = \mathbf{h}\left[\mathbf{x}(t_i), t_i\right] + \mathbf{v}(t_i) \tag{2.30}$$

where $\mathbf{v}(t_i)$ is the same as described in Equation (2.21). The measurement matrix

$$\mathbf{H}\left[t_i; \hat{\mathbf{x}}(t_i^-)\right] = \frac{\delta \mathbf{h}\left[\mathbf{x}, t_i\right]}{\delta \mathbf{x}}\bigg|_{\mathbf{x}=\hat{\mathbf{x}}(t_i^-)} \tag{2.31}$$

is substituted into Equations (2.23) and (2.25) to produce the equations to update the covariance and compute the Kalman gain. The state update equation becomes

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)\left\{\mathbf{z}_i - \mathbf{h}\left[\hat{\mathbf{x}}(t_i^-), t_i\right]\right\} \tag{2.32}$$

where $\mathbf{h}\left[\hat{\mathbf{x}}(t_i^-), t_i\right]$ is now the predicted measurement.

19

### 2.4.2 Residual Monitoring.

Built into the Kalman filter algorithm is the ability to monitor the quality of the measurements the system is receiving. For the linear case the residual

$$\mathbf{r}(t_i) = \mathbf{z}_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-) \tag{2.33}$$

is the difference between the predicted measurement, $\mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)$, and the measurements at a given time step $t_i$. For the Extended Kalman Filter (EKF) the residual is

$$\mathbf{r}(t_i) = \mathbf{z}_i - \mathbf{h}\left[\mathbf{x}(t_i), t_i\right] \tag{2.34}$$

It is shown in [14] that the residual is zero-mean with a covariance of

$$\mathbf{E}\left\{\mathbf{r}(t_i)\mathbf{r}^{\mathbf{T}}(t_i)\right\} = \mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^T(t_i) + \mathbf{R}(t_i) \tag{2.35}$$

By comparing the square root of the diagonal elements of the residual covariance matrix with the incoming measurements, the algorithm can decide whether or not to include the measurement in the update or to reject the measurement and simply propagate the model. The rejection threshold must be carefully chosen. If the threshold is too high, too many erroneous measurements will corrupt the model. When the threshold is too low, too many valid measurements will be rejected, causing an undesirable amount of drift in the model.

## 2.5 Summary

This chapter reviewed some important topics that are used frequently in Chapter 3. A more detailed discussion of the topics can be found in the references. In the next chapter, a navigation algorithm is developed to estimate the vehicles attitude, position, and velocity. Along with the estimation algorithm a feedback controller is designed to keep the vehicle on a constant heading.

# 3    Methodology

T<small>HE</small> goal of this research is to develop an algorithm that determines the heading, position, and velocity of a small air vehicle flying down a hallway in a building. This chapter begins with an overview of the equipment used for development and testing. In Section 3.2, an algorithm is developed that utilizes an EKF to control the heading of the vehicle and estimate its position and velocity in the navigation frame of reference. The final section of this chapter covers the experimental procedures that will be used to conduct the experiments described in Chapter 4.

## 3.1    System Design

The system that is used to develop the indoor navigation algorithm is comprised of an ARDrone quad-rotor helicopter, and a laptop. The ARDrone transmits video imagery from an on-board camera and telemetry data via a wireless network connection to the laptop. The laptop runs the software algorithms that compute the control commands and transmits them over the wireless network link back to the ARDrone.

The company that developed the ARDrone has released an SDK for a desktop computer system that gives a software developer documented access to the video and telemetry data streams [17]. This requires that any algorithms developed for this project be implemented on the laptop rather than on the drone's internal processor. Implementing new software to run on the vehicle's on-board processor is not supported by the manufacturer and would require a significant reverse engineering effort. The additional development work required for implementing an on-board solution is outside the scope of this research.

### 3.1.1    The ARDrone Platform.

The ARDrone is a small lightweight quad-rotor helicopter. The vehicle comes equipped with a forward-looking high definition video camera, a downward-looking video

camera, an ultrasonic altimeter, a barometric altimeter, a magnetometer, and a MEMS based inertial navigation sensor. The on-board microprocessor integrates the data from the inertial navigation sensors, altimeters, and magnetometer to estimate the vehicle's altitude and attitude. The vehicle's velocity over the ground is estimated using visual odometery with the downward-facing camera. All of the telemetry data is combined with the compressed video stream from the forward-looking camera and broadcast over a wireless network link to a smart phone, laptop, or tablet computer. There are two hardware versions of the ARDrone — version two, shown in Figure 3.1, is used for this experiment.

### 3.1.2  *Remote Processing Computer.*

The computer used for the development of the software is an Alienware M17x laptop. This computer is equipped with an Intel Core i7 quad-core processor running at 2.3 GHz, 6 gigabytes of RAM, and a 256 gigabyte solid state disk drive. The operating system is Ubuntu Linux version 12.04, running version 3.20-35 of the Linux kernel. The computer vision library used in this software is OpenCV version 2.4.2. The ARDrone interface libraries are from the ARDrone SDK version 2.0.

## 3.2  Navigation Software Design

The ARDrones internal inertial sensors are based on low cost MEMS accelerometers and gyroscopes. These low cost sensors are not precise enough to provide an accurate estimate of attitude and velocity by themselves. The drone's internal controller employs a set of algorithms to correct for most of the errors induced by these sensors, so that the system can hold a stable hover. One error that the drone's internal controller can not successfully eliminate in an indoor environment is the yaw gyroscope drift. The main job of the navigation software developed for this research is to utilize an external heading reference to correct the error in the yaw gyroscope. The external heading reference used in this research is a vanishing point extracted from images from the drone's forward-
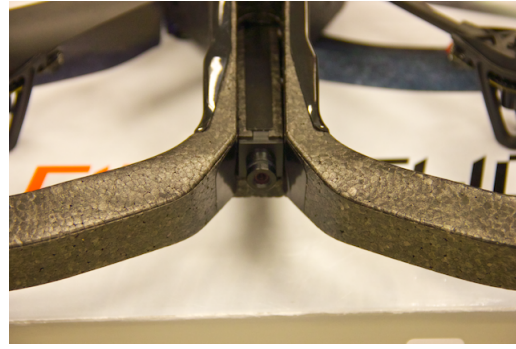
(a)                                          (b)
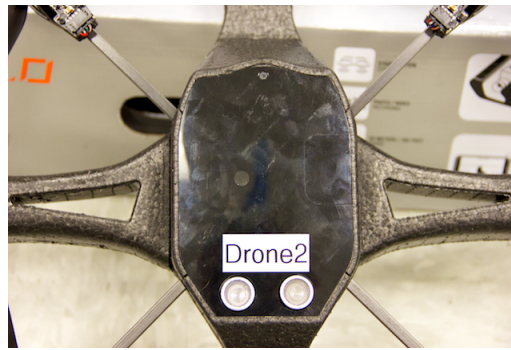


(c)

Figure 3.1: The ARDrone(a) is equipped with a high definition forward-facing video camera(b). The bottom of the vehicle is equipped with an ultrasonic range finder, which appears as the two circles in the bottom of (c). The camera used for visual odometery is located in the small hole near the top of (c).

facing camera. In addition to estimating the correct heading for the vehicle, the navigation software also uses the velocity measurements provided by the drone to estimate the drone's position in the navigation frame.

The navigation software is comprised of three primary components — the vanishing point algorithm, state estimation algorithm, and the heading controller. The software system is shown in Figure 3.2. The functions provided by the SDK are outlined in black. The components developed for this research are shown in the grey shaded box.
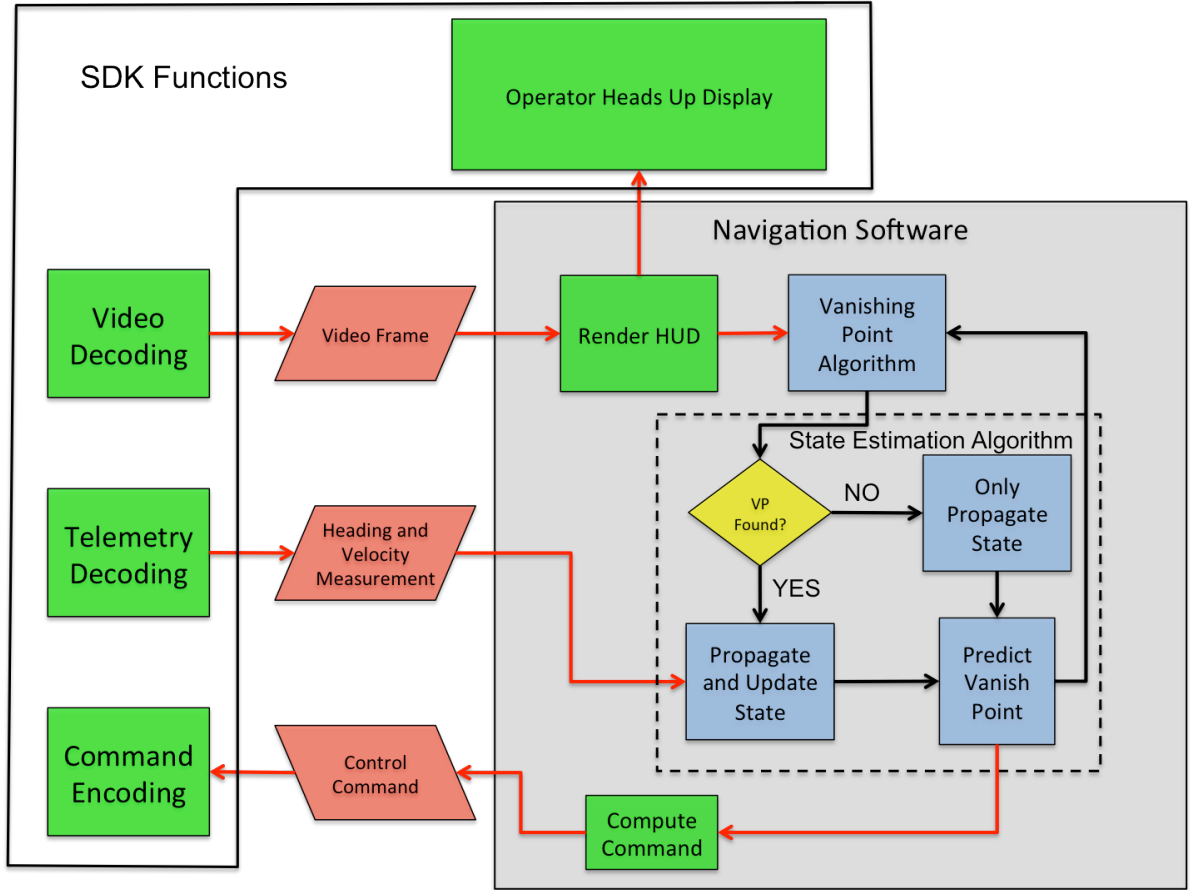
Figure 3.2: This diagram shows the interaction between the navigation software developed for this research and the software supplied by the SDK. The green blocks are separate threads of execution, the blue blocks represent sequential operations. The black lines represent the flow of execution between sequential blocks, and the red lines indicate data flow.

### 3.2.1 The SDK Interface.

One of the programs included with the ARDrone SDK is a ground control station with a graphical user interface (GUI) that provides the user with a live video stream from the drone, displays the telemetry data, and allows the user to command the drone with an external joy stick. The navigation software developed for this research is designed to

integrate into this ground control software. When activated, the navigation software will take over control of the vehicle's yaw rate control channel to steer the nose of the drone in the desired direction.

The ground control software runs a separate thread of execution for each of the basic functions; video decoding, telemetry decoding, command encoding. In addition to these basic functions, other threads are created to handle the display and refresh of the GUI. The threads share data using a shared memory scheme managed by the SDK libraries. The telemetry decoding thread produces updates to the telemetry data at a rate of 200Hz. Each time a new update is ready, it is copied into a buffer controlled by the navigation software. The video decoding works in a similar manner. Video frames arrive at a rate of 30Hz. When a new frame is ready it is passed to the navigation software. The navigation software checks to see if the vanishing point algorithm is ready to process a new frame. If it is, a copy of the frame is sent to the vanishing point algorithm. The navigation software then renders the heads-up display information on the frame and sends it to the GUI to be displayed. The heads-up display information includes the vehicle's position, velocity, heading, and the location of the detected vanishing point.

### 3.2.2   The Vanishing Point Estimation Algorithm.

The job of the vanishing point algorithm is to provide an estimate of the vehicle's heading to the state estimation algorithm described in the next section. The vanishing point estimation algorithm is based on Prahl's work [18]. This section presents the modifications to Prahl's algorithm that are required for this work. A block diagram of the vanishing point algorithm implemented for this system is shown in Figure 3.3. The algorithm begins waiting for a frame to become available from the video decoder. When a new frame is available, it is copied into a buffer, rectified, then passed through a Canny edge detection algorithm. The edge detection algorithm turns the edge pixels in the image white and the non-edge pixels black. This black and white image is then processed by

a line extraction algorithm based on the Hough transform. These algorithms are discussed in detail in[20]. If a predicted vanishing point is available from the state estimation algorithm, the lines that don't pass within five degrees of the predicted vanishing point are discarded. If no vanishing point estimation is available, all of the detected lines are considered candidates. The set of candidate lines is passed to the Random Sample Consensus (RANSAC) algorithm. The RANSAC algorithm is discussed in greater detail in [10]. After the RANSAC algorithm reaches its iteration threshold, the size of each consensus set is examined. The largest consensus set that contains at least three lines is declared the vanishing point. In the case where there is a tie between consensus sets, the point closest to the previous vanishing point is chosen. In the case where no consensus set has a minimum of three lines, an error is returned. The pixel coordinates of the vanishing point are transformed into a unit normal vector in camera frame, and passed to the state estimation algorithm. In the state estimation algorithm, the vanishing point vector is rotated into the navigation frame and the heading is calculated as

$$\Psi_V = \cos(V_y) + \Psi_{offset} \tag{3.1}$$

where $\Psi_V$ is the heading estimated by the vanishing point algorithm, $V_y$ is the $y$ component of the unit normal vector pointing to the vanishing point, and $\Psi_{offset}$ is an offset that is associated with the current vanishing point. The offset is determined based on the current estimate of the heading by

$$\Psi_{offset} = \begin{cases} 0° & -45° \leq (\Psi_R - \delta\psi) \leq 45° \\ 90° & 45° \leq (\Psi_R - \delta\psi) \leq 135° \\ 180° & 135° \leq (\Psi_R - \delta\psi) \leq 180° \\ -90° & -135° \leq (\Psi_R - \delta\psi) \leq -45° \\ -180° & -180° \leq (\Psi_R - \delta\psi) \leq -135° \end{cases} \tag{3.2}$$

26

where $\Psi_R$ is the heading reported by the drone and $\delta\psi$ is the heading error provided by the state estimation algorithm. The need for the offset is discussed further in Section 3.2.2.1.
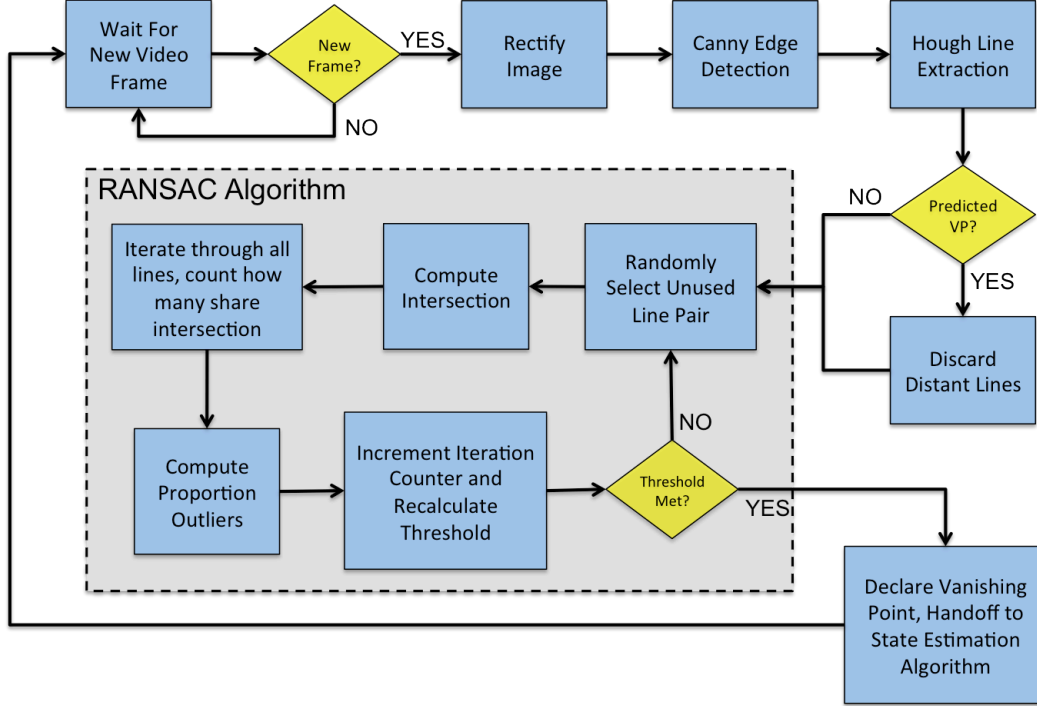


Figure 3.3: A block diagram of the vanishing point detection algorithm.

An example frame from the vanishing point algorithm is shown in Figure 3.4. Here the dark blue lines are the lines chosen by the algorithm as support lines and the green circle is centered over the estimated vanishing point.

### 3.2.2.1  Search Space Reduction.

Prahl's [18] approach estimates the position of a vanishing point for each basis vector in the vehicle's navigation frame of reference. These vanishing points are then used to generate an estimate of the vehicle's orientation. This orientation estimate is then used to generate an estimate of the errors in the inertial navigation system's solution. The three vanishing points that Prahl's code estimates are required to determine the vehicle's
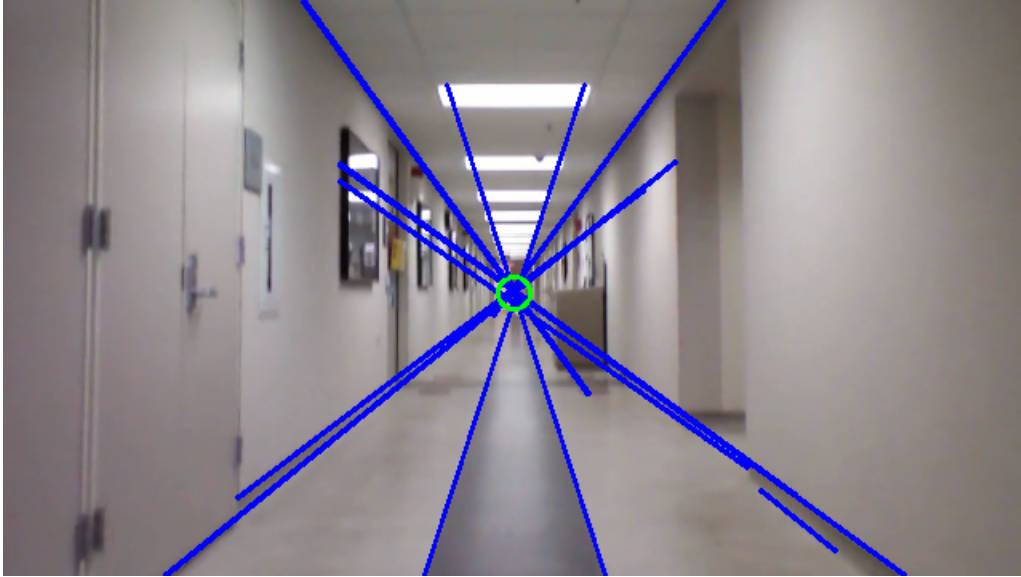
Figure 3.4: A typical image from the vanishing point algorithm. The blue lines are the support lines chosen by the algorithm, the estimated vanishing point is in the center of the green circle.

three Euler angles; yaw, pitch and roll. Because the only information required by the state estimation algorithm to constrain the yaw error is the heading error, only a single vanishing point needs to be located. When a vanishing point is found, the basis vector associated with that vanishing point is ambiguous. This ambiguity is solved during initialization by defining +*X* basis vector based on the location of the first vanishing point found.

A consequence of reducing the search space to a single vanishing point can be observed when the vehicle is rotated through 360°. When tracking a vanishing point for each of the three basis vectors of a reference frame, the heading can be computed relative to one of the basis vectors throughout the entire rotation. If a single vanishing point is being tracked, the heading is measured relative to the basis vector with which the current vanishing point is associated. To compute an absolute heading, an assumption needs to be made about which basis vector the current vanishing point is associated with, so that an

appropriate bias can be added to the relative measurement. The software handles this by using the current heading estimate. When the vehicle rotates past a threshold, the basis vector the detected vanishing points are associated with changes to the next basis vector. A diagram of the thresholds used is shown in Figure 3.5. The thresholds used are spaced at 90° increments starting at 45°. A margin of ±5° is used to add hysteresis. For example, if the vehicle begins with a heading of zero degrees, the detected vanishing point is associated with the +X basis vector. As the vehicle rotates clockwise through a heading of 50°, the basis vector the vanishing points are associated with switches from +X to +Y. The heading measurements are then made relative to 90°.
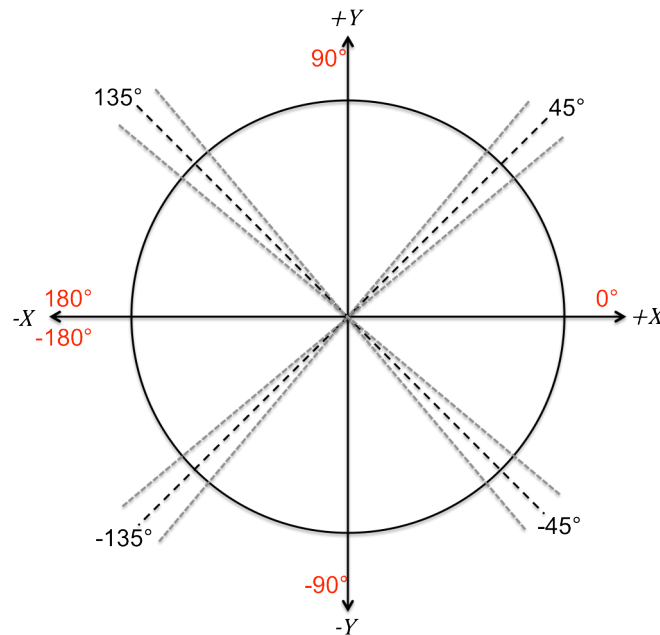


Figure 3.5: The solid black lines represent the basis vectors. The dashed black line is the switch over threshold. The dashed grey lines represent the five degree margin around each threshold. The red numbers are the biases used to calculate the absolute heading.

### *3.2.2.2   Vanishing Point Optimization.*

The second modification made to Prahl's code is a simplification of the method for determining the final vanishing point estimate. In Prahl's implementation, once a set of supporting lines was identified, a gradient descent optimization algorithm was used determine the point with the minimum squared error relative to the set of support lines. In order to reduce the computational complexity, the gradient descent algorithm is removed. Instead, the point with the largest consensus set is returned as the vanishing point. In the case where multiple points have equally sized consensus sets, the point that is closest to the predicted vanishing point is returned as the vanishing point. Removing the gradient descent optimization is expected to increase the measurement noise; however, the high measurement rate coupled with residual monitoring is expected to make the effect of this change negligible.

### *3.2.3   The State Estimation Algorithm.*

The state estimation algorithm is the core of the navigation software. The algorithm utilizes an EKF to estimate the position, heading and velocity of the vehicle based on heading and velocity measurements. The following sections develop a minimal system dynamics model, and measurement model used by the EKF algorithm.

### *3.2.3.1   The ARDrone System Model.*

The ARDrone contains an internal stabilization system that makes developing a precise model of the system's dynamics difficult, because the observable output from the system is not a linear function of the input to the system. Because the time between measurements for this system is very small relative to the vehicle's velocity, the input to the system model can be neglected. The system model developed in this section is a stochastic linear system which takes the form of

$$\dot{\mathbf{x}}(t) = \mathbf{F}(t)\mathbf{x}(t) + \mathbf{G}(t)\mathbf{w}(t) \tag{3.3}$$

where $\mathbf{x}(t)$ is the state vector, and $\mathbf{w}(t)$ is a vector of zero-mean, white, Gaussian random processes. The matrices $\mathbf{F}(t)$, and $\mathbf{G}(t)$ contain coefficients that relate the terms in the state vector to the terms in the process noise vector.

The vehicle provides velocity measurements relative to measurement frame of reference and a heading measurement relative to magnetic north. The heading estimate is modeled as

$$\Psi_R = \Psi_{True} + \delta\psi \tag{3.4}$$

where $\Psi_R$ is the reported heading, $\Psi_{True}$ is the true heading and $\delta\psi$ is some unknown error. Because the input to the system is not included in the system model, only the heading error is tracked by the filter. To determine an appropriate model for the heading error, a preliminary experiment was conducted with an external truth system. The external truth system used was a Vicon system, which uses high-speed cameras to track the position of optical markers attached to the vehicle. In this experiment, the vehicle was commanded to fly in a straight line along a course approximately 30 feet in length. The difference between the vehicle's reported heading and the heading measured by the Vicon system is used as the heading error. An example of this error is shown in Figure 3.6. This plot shows that the heading error contains noise and a time dependent bias. In order to properly model the heading error, an additional state for the heading error rate is required.

The model used to relate the velocity measurements to the position of the vehicle in the navigation frame is

$$X^N = \int_{t_0}^{t_n} V_{x_{meas}} cos(\Psi_R(t) - \delta\psi(t)) - V_{y_{meas}} sin(\Psi_R(t) - \delta\psi(t)) dt \tag{3.5}$$

$$Y^N = \int_{t_0}^{t_n} V_{x_{meas}} sin(\Psi_R(t) - \delta\psi(t)) + V_{y_{meas}} cos(\Psi_R(t) - \delta\psi(t)) dt \tag{3.6}$$

where $V_{x_{meas}}$ and $V_{y_{meas}}$ are the velocity measurements, and the terms $X^N$ and $Y^N$ are the position states.

Combining these models into a system of equations results in the state vector and state transition matrix. The state vector is defined as

$$\mathbf{x}(t) = \begin{bmatrix} X^N & Y^N & \delta\psi & \dot{\delta\psi} & V_x^N & V_y^N \end{bmatrix}^T \tag{3.7}$$

where $X^N$ and $Y^N$ represent the position in the navigation frame of reference, $\delta\psi$ represents the heading error, $\dot{\delta\psi}$ represents the heading error rate, and $V_x^N$ and $V_y^N$ represent the velocity in the navigation frame of reference. The heading error and heading error rate are measured in radians, the velocity states are measured in meters per second, and the position is measured in meters. The coefficients in the state transition matrix

$$\mathbf{F}(t) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.8}$$

relate the velocity states to the position states and the heading error rate state to the heading error state.

In addition to the system dynamic model, a model for the process noise for each sate is required. For this system, the process noise for the heading error, heading error rate, and the velocity states are modeled as random walk. Because the position states are derived from the velocity states, no process noise is included for the position states. The resulting noise vector is

$$\mathbf{w}(t) = \begin{bmatrix} w_{\delta\psi}(t) \\ w_{\dot{\delta\psi}}(t) \\ w_{V_X^N}(t) \\ w_{V_Y^N}(t) \end{bmatrix} \tag{3.9}$$

32
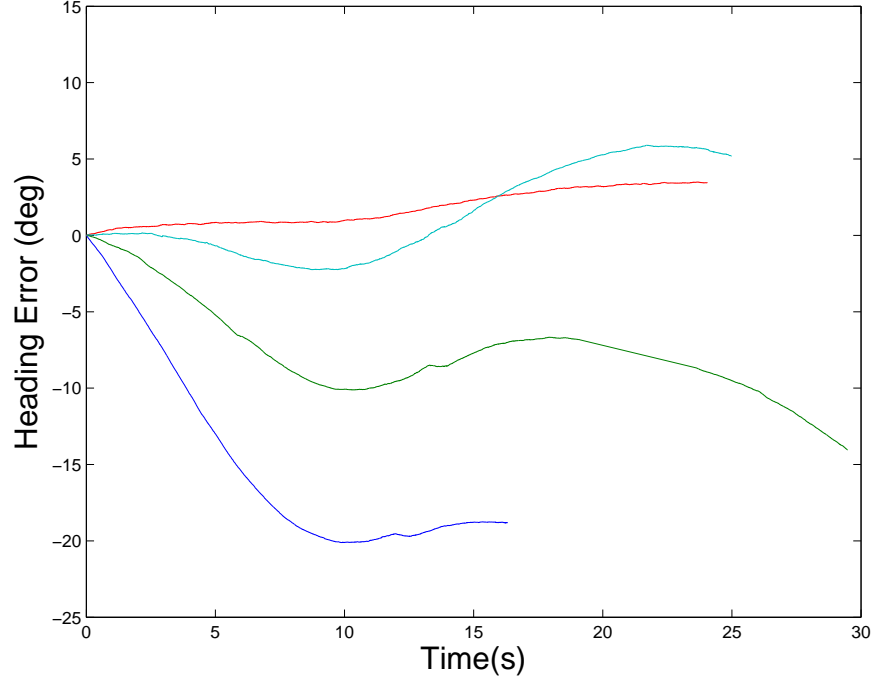
Figure 3.6: The heading error is characterized by flying the drone in a straight line and taking the difference between heading measured by the Vicon system and the drone's reported heading. The heading error of the drone in each flight has a trend which is indicative of a rate error.

where $w_{\delta\psi}(t)$ represents the heading error process noise, $w_{\dot{\delta\psi}}(t)$ represents the heading error rate process noise. The terms $w_{V_X^N}$ and $w_{V_Y^N}(t)$ represent the process noise on the velocity states. The coefficient matrix

$$\mathbf{G}(t) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

33

relates the noise terms to the appropriate states. The strength of each noise process represents the amount of uncertainty in the system due to factors that are not included in the model. The initial values for the noise strength are shown here in the noise covariance matrix

$$\mathbf{Q}(t) = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.0625 & 0 \\ 0 & 0 & 0 & 0.0625 \end{bmatrix} \tag{3.11}$$

where the diagonal terms are the covariance of the corresponding noise terms. One consequence of not modeling user input to the system is the user input must be treated as increased uncertainty in the system. These initial values in $\mathbf{Q}(t)$ were chosen to account for this additional uncertainty and will be further investigated in Chapter 4.

### 3.2.3.2 *Kalman Filter Design.*

The EKF algorithm, presented in Section 2.4, provides an estimation of the values of the states based on the system dynamic model and the available measurements. The state propagation equations require the computation of the matrix exponential function, which has a computational complexity of $O(n^3)$ [7]. If the rate the system changes is small compared to the rate at which measurements are available, the first order approximations of the propagation equations can be used. Using the approximations is desirable for this system because it reduces the computational complexity to $O(n)$.

The measurements are driven by the vanishing point algorithm. Each time it completes a cycle the most recent telemetry data is used to propagate and update the states. If no vanishing point is found, the states are only propagated. The stochastic nature of the vanishing point algorithm means that measurements are not taken on a fixed interval. The histogram in Figure 3.7 shows a typical distribution of the time it takes for the vanishing point algorithm to complete. The majority of the time, it takes between 30 and 40 milliseconds to complete. At this time interval, the difference between the full solution and

the first order approximation is on the order of $10^{-17}$, which is negligible for this system. It is important to note that even though the velocity measurements are available from the telemetry process every five milliseconds, the velocity states are not updated at this rate. They are updated at the same time as the heading states, using the most recent velocity measurements.
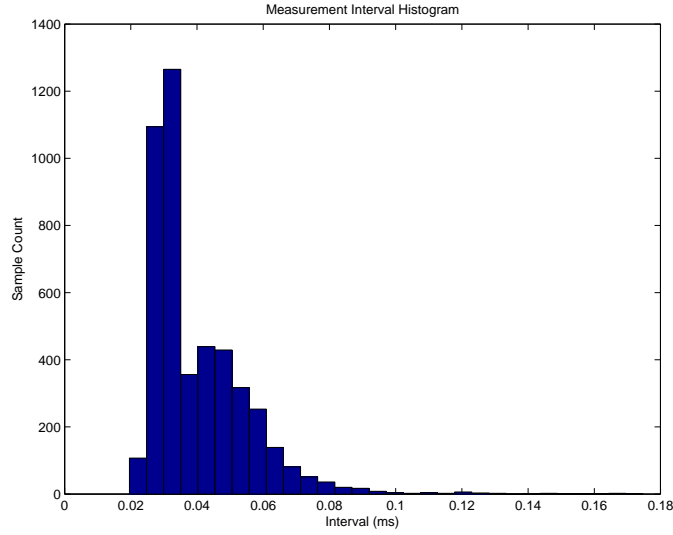


Figure 3.7: This histogram depicts a typical statistical distribution of the measurement time interval. The majority of measurements are taken at an interval of between 30 and 40 milliseconds.

With the system model defined in the previous section, a model that relates the measurements to the states is required. The heading error measurement

$$\Psi_R - \Psi_V = \delta\psi + v_1(t) \tag{3.12}$$

where $\Psi_R$ is the same reported heading used in the system model, $\Psi_V$ is the heading estimated by the vanishing point algorithm, $\delta\psi$ is the heading error, and $v_1(t)$ is a zero-mean, white, Gaussian noise process.

One of the problems with the velocity measurements is that they are heavily dependent on the patterns of the surface over which the drone is flying. Reliable measurements can only be made over surfaces with a high contrast pattern of the appropriate scale. Because of the low reliability of the velocity measurements, only a basic model of the velocity measurements is developed for this system. The velocity measurements are modeled as

$$V_{x_{meas}}(t) = V_x^N(t)\cos(\Psi_R - \delta\psi(t)) + V_y^N(t)\sin(\Psi_R - \delta\psi(t)) + v_2(t) \qquad (3.13)$$

$$V_{y_{meas}}(t) = -V_x^N(t)\sin(\Psi_R - \delta\psi(t)) + V_y^N(t)\cos(\Psi_R - \delta\psi(t)) + v_3(t) \qquad (3.14)$$

where the terms $V_{x_{meas}}$ and $V_{y_{meas}}$ represent the velocity measurements from the drone, and the terms $V_x^N$ and $V_y^N$ are the velocity states. The $v_2(t)$ and $v_3(t)$ terms are zero-mean, white, Gaussian noise processes. These equations are used to build the measurement matrix

$$\mathbf{H}\left[\hat{\mathbf{x}}(t_i^-), t\right] = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \chi_x & 0 & \cos(\Psi_R - \delta\psi(t_i^-)) & \sin(\Psi_R - \delta\psi(t_i^-)) \\ 0 & 0 & \chi_y & 0 & -\sin(\Psi_R - \delta\psi(t_i^-)) & \cos(\Psi_R - \delta\psi(t_i^-)) \end{bmatrix} \qquad (3.15)$$

The quantities $\chi_x$ and $\chi_y$ represent the solutions to

$$\chi_x = \frac{d}{d\delta\psi}\left(V_x^N(t)\cos(\Psi_R - \delta\psi(t)) + V_y^N\sin(\Psi_R - \delta\psi(t))\right)\bigg|_{\mathbf{x}=\hat{\mathbf{x}}(t_i^-)} \qquad (3.16)$$

$$\chi_y = \frac{d}{d\delta\psi}\left(-V_x^N(t)\sin(\Psi_R - \delta\psi(t)) + V_y^N\cos(\Psi_R - \delta\psi(t))\right)\bigg|_{\mathbf{x}=\hat{\mathbf{x}}(t_i^-)} \qquad (3.17)$$

which present a design option. These terms feed the uncertainty in the heading error into the uncertainty in the velocity. When the uncertainty in the heading error is much smaller than the uncertainty in the velocity, the effect these terms have is negligible. For this system these terms were found to be negligible, but were included here for completeness.

The drones report velocity in both the *X* and *Y* directions. For this system, the noise on the *X* velocity measurement is assumed to be a sufficient estimate of the noise in the *Y* direction. The characterization of the measurement noise of the velocity measurements is done using the *X* velocity measurements recorded over a total of nine flights. The *X* velocity

is examined to determine the point where the vehicle reaches a steady state velocity. The steady state velocity measurements are extracted from the data set and processed to remove the mean velocity. The histogram in Figure 3.8 shows a typical distribution of the zero-mean velocity measurements. The distribution appears to be approximately Gaussian with a variance of 0.0064 m/s. This approach generates an upper limit on the measurement noise, because the effects of the environment can not be removed. This approach also assumes that there is no constant bias in the velocity measurements.
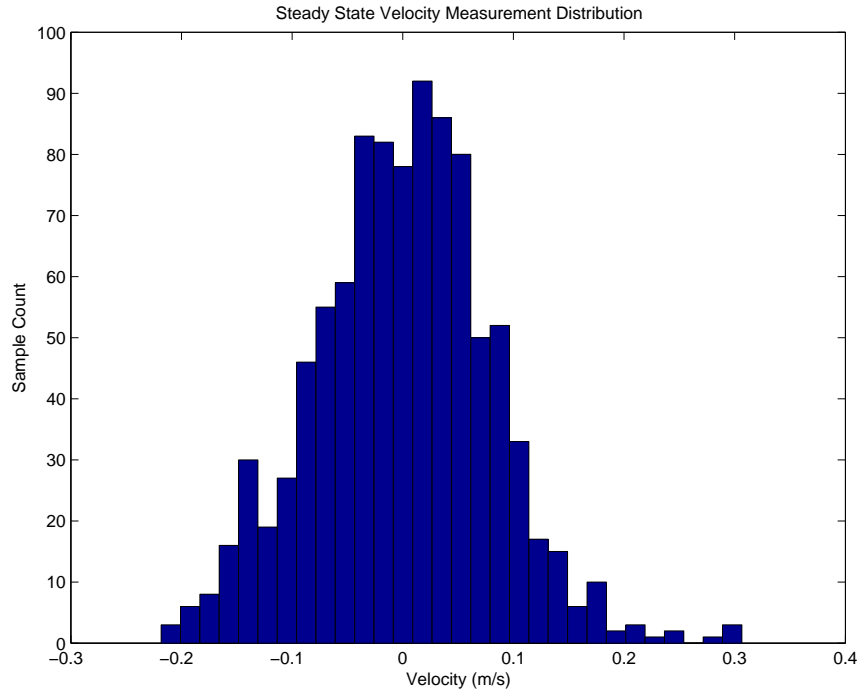


Figure 3.8: The histogram shows the distribution of velocity measurements taken by the three drones over nine flights. Each flight's data was adjusted to remove the mean velocity.

Characterization of the measurement noise of the vanishing point algorithm is difficult. When flying, the drone uses the current estimate of the heading error state to predict the location of the vanishing point, and restricts the search for the vanishing point to a fixed 5° area around this estimate. When the drone is setup on a static mount with a constant zero

heading, the scene doesn't change. This causes the same lines to be chosen to estimate the vanishing point, making the measurement noise lower than it would be in flight. For this system, a recursive approach was used to determine the initial estimate of the measurement noise. First, an initial guess of the measurement noise was based on the results of a static test. Then system was flown down a hallway with an approximately constant heading. The resulting vanishing point heading measurements from the flight test are summarized in the histogram shown in Figure 3.9. The distribution of the measurements is modeled as a zero-mean, white, Gaussian process with a standard deviation of about one degree. The heading measurement from the vanishing point algorithm is a relative measurement, therefore measurement noise is not a function of the vanishing point being tracked. When the vehicle rotates and begins tracking a new vanishing point, a constant bias is added to the relative heading measurement produced by the vanishing point algorithm. The initial measurement covariance matrix used for this system is

$$\mathbf{R}(t_i) = \begin{bmatrix} 0.0016 & 0 & 0 \\ 0 & 0.0064 & 0 \\ 0 & 0 & 0.0064 \end{bmatrix} \tag{3.18}$$

where the diagonal terms are the covariances of the measurement noises. The measurement noise for the heading error is in radians squared and the noise for the velocity measurements is measured in meters per second squared. The off diagonal terms of the covariance matrix are left zero. There is cross correlation between the heading error measurement noise and the velocity measurement noise, but it is assumed to be negligible. The heading measurement returned by the vanishing point algorithm, described in Section 3.2.2, is calculated based on a point in space that is assumed to be the vanishing point. The algorithm defines this point as the point in the scene with the largest consensus set, or set of lines that pass through it. A majority of the time, the point in the scene with the largest consensus set is the true vanishing point; however, it is possible that there is a point on the

38
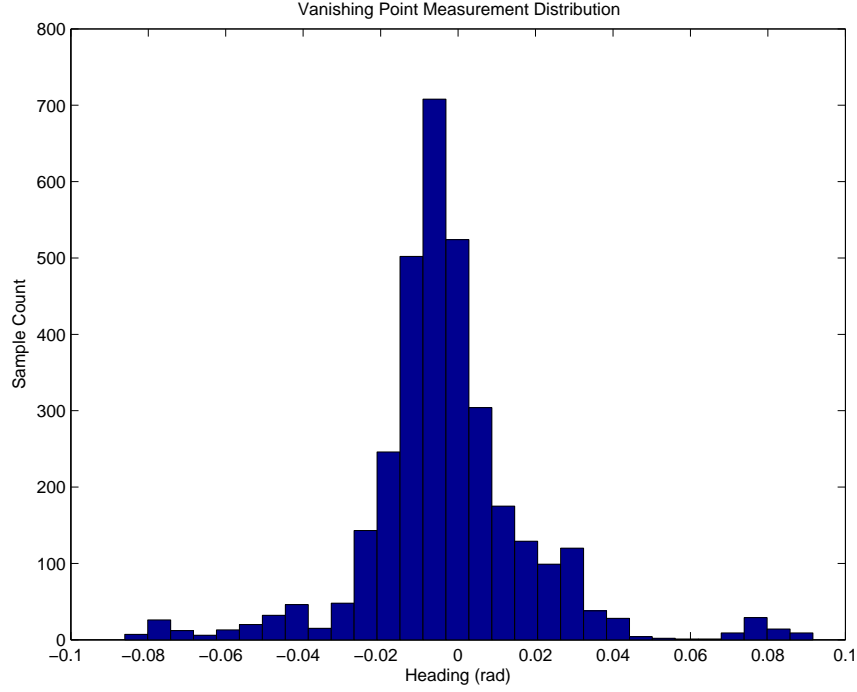
Figure 3.9: The distribution of the heading measurement errors. This distribution is modeled as a zero-mean, Gaussian distribution with a standard deviation of approximately 1 degree.

scene that has a greater consensus set than the actual vanishing point. The system needs a method for identifying and rejecting heading measurements based on these false vanishing points. Failure to do so will cause the system to incorrectly predict the location of the vanishing point for the next search cycle. The result is a system that will either fail to find a vanishing point, or worse, track a false vanishing point. The navigation software does this by monitoring performing residual monitoring as discussed in [14]. Residual monitoring provides a method for checking each measurement prior to integrating the measurement into the system states. For this work, a rejection threshold of $1.5\sigma$ is used. This will cause approximately 13% of the valid measurements to be rejected. As a result, the average

measurement rate increases to 45 milliseconds from the observed 40 milliseconds. The slower measurement rate is still sufficient for this research.

### 3.2.4  Heading Controller.

The final component of the navigation software is a proportional control loop used to maintain a constant heading. When activated, a control signal is generated to drive the heading error to zero. In this case the heading error is

$$\Delta\Psi = \Psi_{cmd} - \Psi_R - \delta\psi \tag{3.19}$$

where $\Delta\Psi$ is the control error, and $\Psi_{cmd}$ is the commanded heading. The vehicle's input commands are required to be between negative one and one, as a result the proportional control equation becomes

$$S_{\dot\psi}(t) = \begin{cases} -1 & K_p\Delta\Psi < -1 \\ K_p\Delta\Psi & -1 \le K_p\Delta\Psi \le 1 \\ 1 & K_p\Delta\psi > 1 \end{cases} \tag{3.20}$$

where $K_p$ is the proportional control constant. The value for the proportional control constant is chosen to provide a reasonable response rate with minimal overshoot. For this work $K_p = 0.01$ was found to meet this criteria.

## 3.3  Procedures

There are some basic procedures that are common to all of the experiments described in the following sections. This section contains a brief summary of the procedures for calibrating the cameras, collecting the data, and performing the flight tests.

### 3.3.1  Camera Calibration.

For each drone used, a camera calibration set for the forward facing camera is collected. Each calibration set consists of a total of 27 images collected from nine different positions in front of a standard calibration board. The same calibration board is used for

each of the drones. The camera calibration toolbox [5] for MATLAB, is used to determine the values of the camera matrix and distortion coefficients for each of the drone's cameras. This procedure is only conducted once prior to running the experiments developed in Chapter 4.

### 3.3.2 Data Collection.

Data for each experiment is collected by the navigation software. A button on the flight controller is used to start and stop recording to the data file. During a typical collection, recording is started, the takeoff command is issued, a state reset command is issued, then the flight begins. When the flight plan is complete, data recording is turned off before the land command is issued. The collected data is stored in a comma separated value (CSV) file for later processing in MATLAB.

When power is first connected to the ARDrone, the heading is initialized to zero degrees. Just after the takeoff sequence completes, there is a large jump in the heading reported by the drone. The state estimation algorithm is not designed to handle large steps in heading, so these large steps cause large errors in the estimated heading. The exact cause of this heading step is unknown. By examining the telemetry data it appears that after the vehicle completes its takeoff sequence, it takes a reading from the on-board magnetometer to compute a heading offset from magnetic north. In order to mitigate these heading steps, the state values are reinitialized after the vehicle is airborne. After the states have successfully been reset, the heading controller is engaged. During experiments that require slewing between vanishing points, the heading controller is disengaged so the vehicle can be manually rotated.

### 3.3.3 Flight Profile.

Each flight is conducted at a target altitude of 1 meter and a desired forward velocity between two and three meters per second. The ARDrone's internal controller attempts to maintain a constant altitude, but it has been observed to vary by as much as half a meter.

The drone's speed is controlled by the commanded pitch. For a given negative pitch angle the vehicle will undergo forward acceleration until the aerodynamic forces equalizes the vehicle's speed. Errors between the commanded pitch and the pitch achieved by the drone have been observed to result in speed variances of up to 0.5 meters per second.

## 3.4   Summary

The first section of this chapter presented the design of the navigation algorithms, the system model, and the experimental setup.   After documenting the experimental procedures, the experiments for this research are developed.   The resulting data will be analyzed in the following chapter.

# 4    Results

Iɴ Chapter 3, a navigation algorithm is developed to control the heading on the ARDrone and estimate its position and velocity.  This chapter presents a set of experiments designed to validate the navigation algorithm's design and characterize its performance. The first set of experiments are a series of flight tests designed to validate the vanishing point estimation algorithm and Kalman filter design.  These flight tests are conducted in a building and the algorithms are run in real time.  In the second set of experiments, a MATLAB program will be used to reprocess the measurement data recorded during the flight tests to examine in detail the settings chosen for the Kalman filter. During the post-processing experiments, an error in the real-time software was discovered.  An additional post-processing experiment was developed to determine the effects of this software error.

## 4.1    Constant Heading Experiment

The objective of this test was to demonstrate the ability of the drone to maintain a constant heading while traveling along a straight path. This test utilized the course depicted in Figure 4.1.  The vehicle ran the course three times in the $+X$ direction.  The hallway utilized in this test is pictured in Figure 4.2. Preliminary experiments found that the texture in the floor was not of the appropriate scale to provide consistent velocity measurements. Examples of the original floor texture are shown in Figure 4.3.  In order to improve the consistency of the velocity measurements, strips of tape were used to add more texture to the floor. The interval for the tape strips is such that at least two pieces of tape are in the downward-looking camera's field of view at any given point, but are not placed at a measured interval.
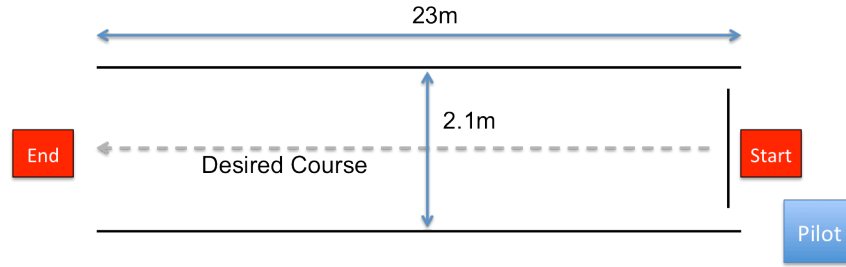
Figure 4.1: Test course used to demonstrate constant heading and position determination.



Figure 4.2: The hallway used for both the constant heading and vanishing point transition tests. The pieces of tape were added to the floor to improve the velocity measurements.

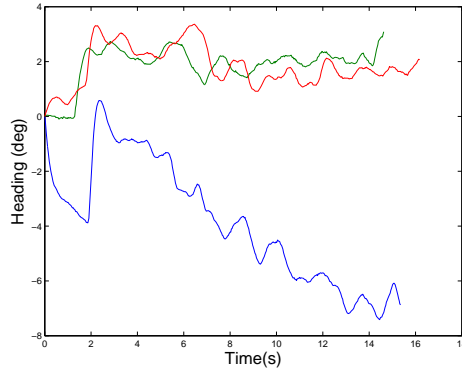### 4.1.1  Constant Heading Results.

The data presented in this section is from Drone 1, but it is representative of the performance of all of the drones. The heading measurement reported by the ARDrone is shown in Figure 4.4a. The ground track in Figure 4.4b was created with a MATLAB program using the ARDrone's heading and velocity measurements. The location of the

Figure 4.3: Closeup images of the hallway floor. As can be seen the detail in the floor is inconsistent along the course, and does not provide regular enough features for the velocity measurement system.

walls of the hallway are represented by the black lines. The heading estimated by the navigation software is shown in Figure 4.4a. The estimated position of the vehicle in the navigation frame, tracked by the position states, is shown in 4.4c. Of particular interest in Figure 4.4b, is the simulated ground track of the first flight. It is counterintuitive that the flight with the largest heading error tracked farther down the hall before striking a simulated wall than the other two flights with relatively small heading errors. This anomaly is due to a combination of errors in the velocity and heading measurements on-board the drone. A small negative heading error would have the effect of compensating for a velocity bias in the $+Y$ direction. However, without external truth data it is not possible to determine the exact cause of the error. With the heading controller engaged, the vehicle maintains a heading within one degree of the desired heading as shown in 4.4c. The slight drift from zero can be attributed to the lack of integral control in the heading controller. When the heading error is less than one degree, the resulting control input small enough that it is ignored by the vehicle.
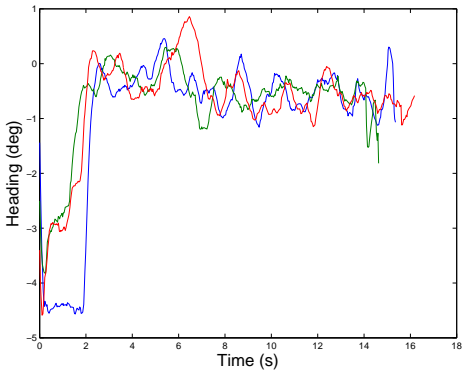
The heading error state from the Kalman filter is shown in Figure 4.5. This state is subtracted from the ARDrone's measured heading to produce a heading of 0°, therefore it is
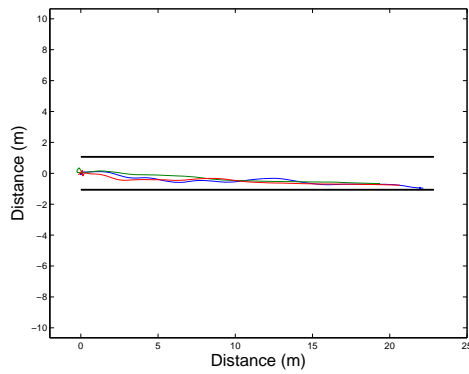
(a) ARDrone Heading Measurement
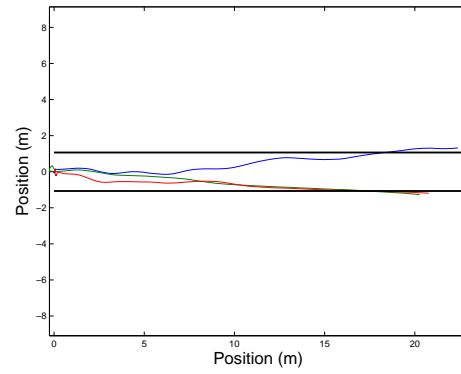
(b) Reconstructed Ground Track

(c) Estimated Heading

(d) Estimated Ground Track

Figure 4.4: A sample of the typical results from the constant heading experiments. This data represents three flights from Drone 1. The ARDrones heading measurements display a varying rate bias(a), which effects the reconstructed ground track(b). The heading estimated by the Kalman filter allows the drone to stay on a course with in 1° of the desired heading 0° (c), and produces a more consistent ground track(d).

expected to follow the same trend as the ARDrone heading measurements shown in Figure 4.4a. The heading error state residual is shown in Figure 4.6. As expected the residual does not diverge and is roughly zero mean. The dotted lines represent the residual monitoring threshold of $\pm 1.5\sigma$. It is expected that approximately 13% of the measurements should lie outside the residual threshold; however, here this isn't the case. When the initial Kalman

filter settings were chosen in Chapter 3, the process and measurement noise were estimated to be larger than they were in reality. The overestimation is required to compensate for the greater uncertainty in the model caused by user input. The effects of these settings are discussed further in Section 4.3.3.
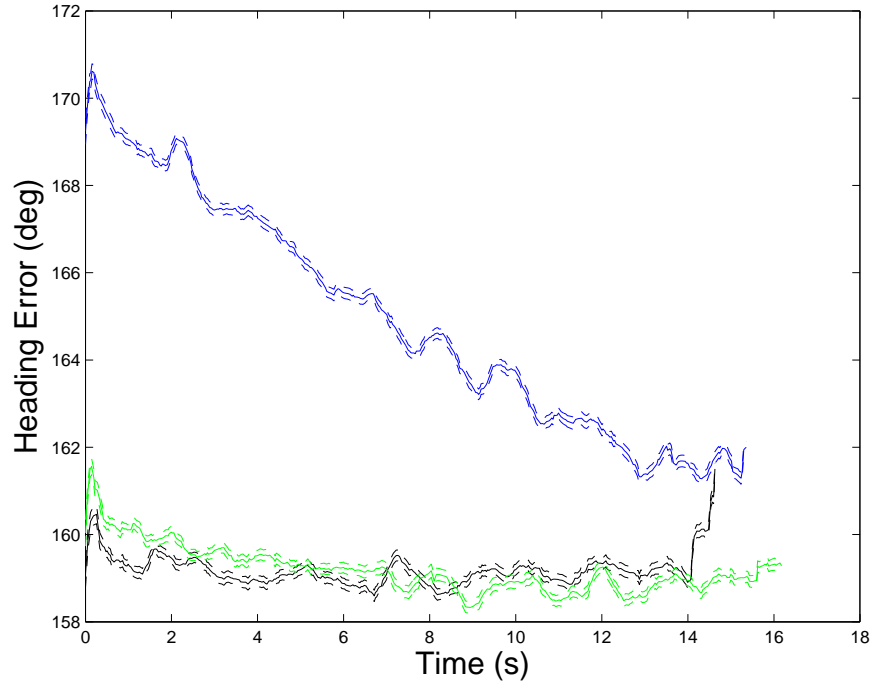


Figure 4.5: Typical results for the heading error state during the constant heading experiment. This data was taken from three flights of Drone 1. The dashed lines represent $\pm 1\sigma$.

The velocity states are shown in Figures 4.7a and 4.7b. As expected, the vehicle accelerates until it reaches a nearly steady-state velocity. The velocity measurement residual is shown in Figures 4.7c and 4.7d. Again, it is apparent that the strength of the process and/or measurement noise is larger than what would be expected.
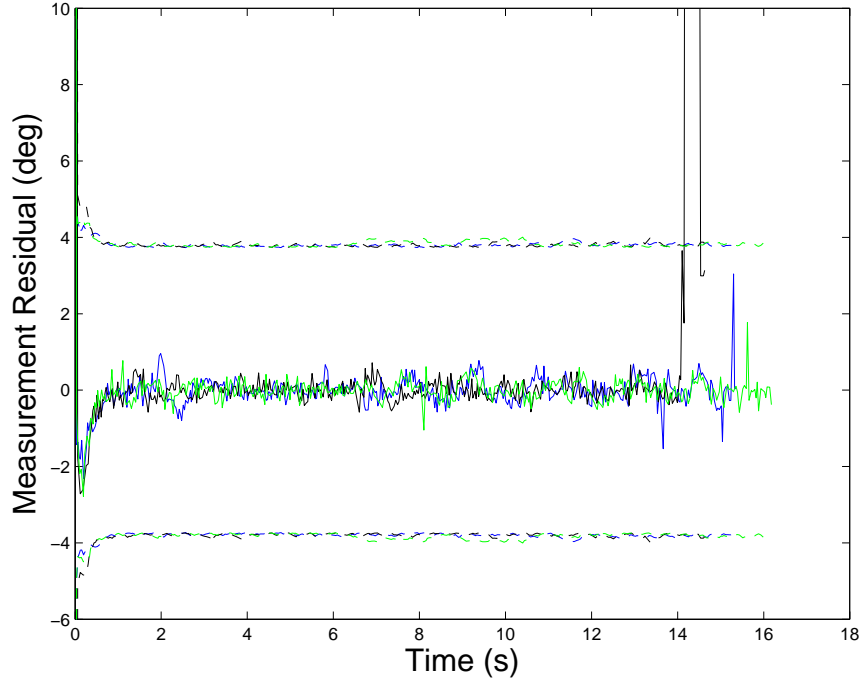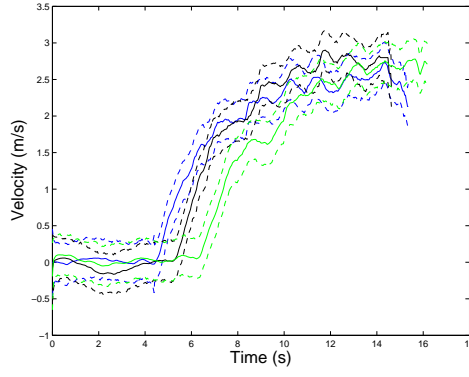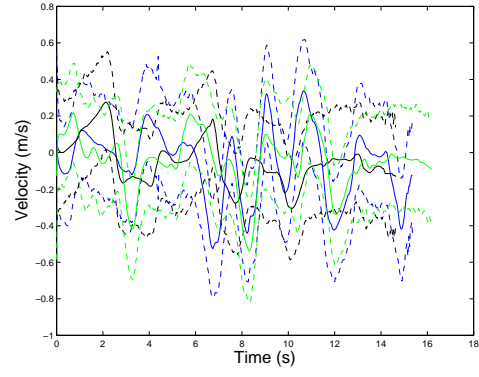
47

Figure 4.6: Typical measurement residual for the heading error state from the constant heading experiment. This data was taken from three flights of Drone 1 during the constant heading experiment. The solid lines represent the measurement residual and the dashed lines represent the $\pm 1.5\sigma$ residual rejection threshold.

## 4.2 Vanishing Point Transition Experiment

The objective of this test is to demonstrate the vehicle's ability to travel for a period of time tracking its initial vanishing point, then rotate 90° and begin tracking a new vanishing point. For this test, each vehicle traveled the course three times in each direction. Note that when the navigation algorithm initializes, it defines the first vanishing point it finds as the one in the +X direction. The course layout for this test is depicted in Figure 4.8. This course is set up utilizing the same hallway as the previous experiment for one leg and the second leg of the course is laid out in an intersecting hall with similar geometry to the

(a) $X^N$ Velocity State

(b) $Y^N$ Velocity State
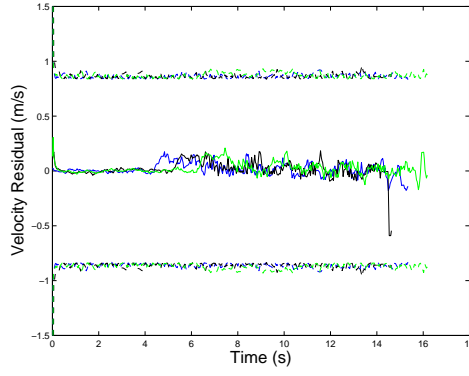
(c) $X_{MEAS}$ Velocity Measurement Residual
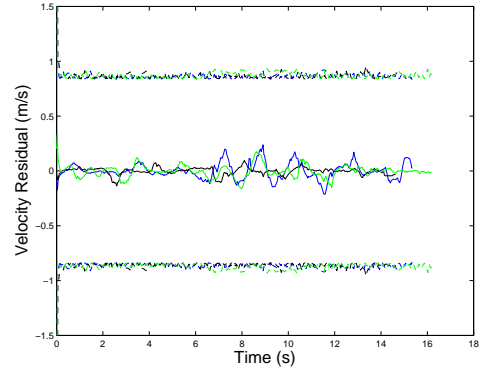
(d) $Y_{MEAS}$ Velocity Measurement Residual

Figure 4.7: Typical results for the $X$(a) and $Y$(b) velocity states during the constant heading experiment. The dashed lines represent $\pm 1\sigma$. The measurement residuals are shown in (c) and (d), where the dashed lines represent the $\pm 1.5\sigma$ residual rejection threshold. This data was taken from three flights of Drone 3, and is representative of the results from the other two drones.

first leg. Tape is placed on the floor in the second leg as well to ensure consistent speed measurements for both legs of the course.

Figure 4.8: The test course used to demonstrate tracking handoff between vanishing points.

### 4.2.1   Vanishing Point Transition Experiment Results.

As in the previous test, the results were consistent across all three vehicles. The unaided heading plot is shown in Figure 4.10a. Examining the raw he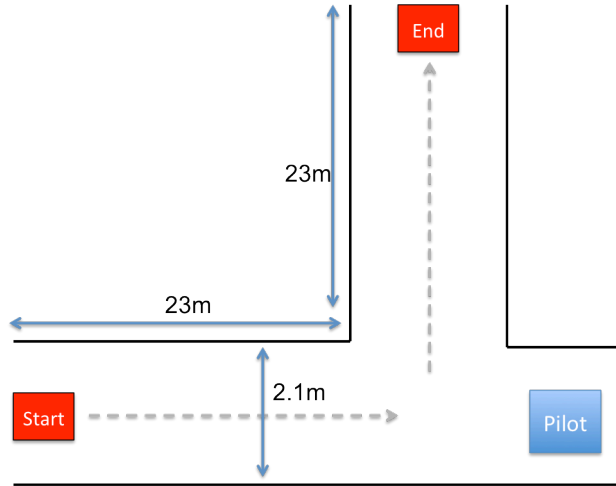ading data, reveals an interesting error in the drones on-board navigation system shown in Figure 4.9. Drones 1 and 3 appear to be able to maintain a reasonably constant heading estimate for the first leg of the test course. However, during the second leg, a heading rate bias appears. This rate bias is not consistent across the runs or the drones. Drone 2 has a almost the opposite problem. It begins the course with a constant rate bias which disappears after the 90° turn. The rest of the data presented here is from vehicle number three.

The the ground tracks reconstructed from the drone's measurements are shown in Figure 4.10b. The solid black lines represent the boundaries of the test hallway. The position estimation errors become more obvious during this test. It is clear that without supplemental heading information, the vehicle would not be capable of navigating this relatively simple course based solely measurements from the drone's navigation system.
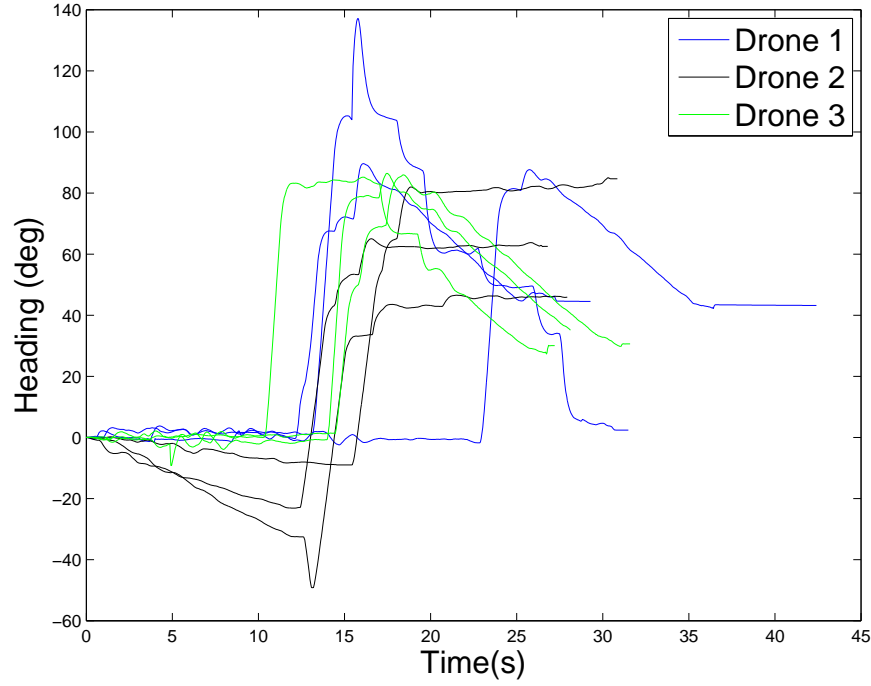
50

Figure 4.9: The heading measurements from the navigation system on all three drones.

#### 4.2.1.1 Heading Error State.

Typical results for the heading error state are shown in Figure 4.11a. During the first leg of the flight, the error state tracks near zero, compensating for a steady state bias. After the 90° turn, the error state tracks the induced heading rate through the end of the flight. The heading error residual is shown in Figure 4.11b. An example of the typical behavior

of the heading error state during the vanishing point transition is shown in Figure 4.12a. As expected when the measurement of the heading error jumps rapidly, the residual monitoring effectively rejects the measurements as errors. This can also be seen in the measurement residual shown in Figure 4.12b. This data was collected with the same noise settings as the previous experiment in Section 4.1, so it is expected that the heading error state covariance would begin to grow rapidly while the measurements are unavailable. The cause of this anomaly is due to an error in the navigation algorithm which will be investigated further in

(a) Reported Heading

(b) Reconstructed Ground Track
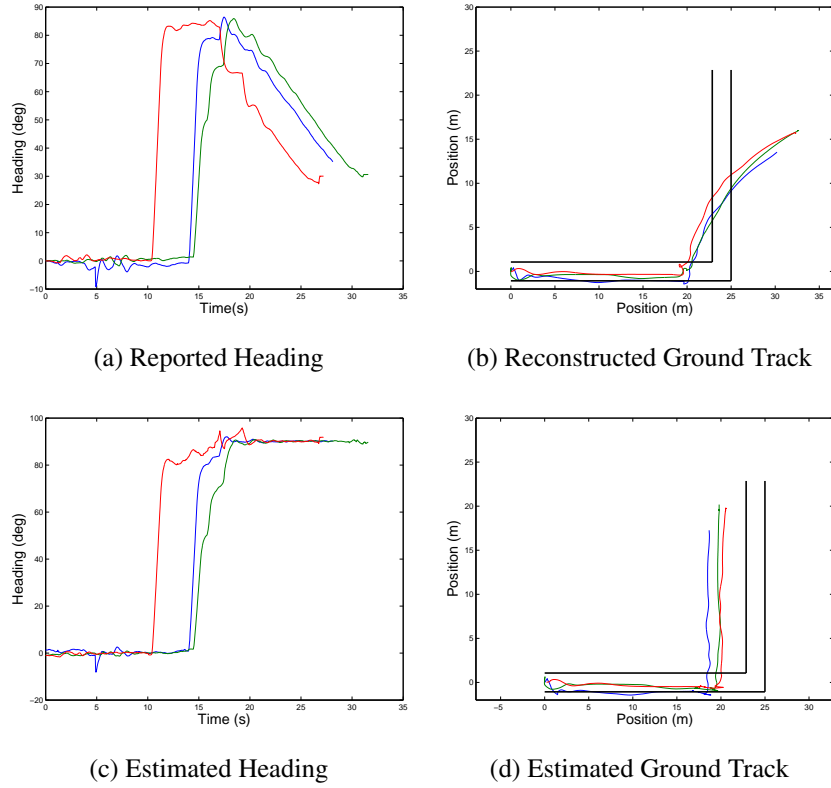
(c) Estimated Heading

(d) Estimated Ground Track

Figure 4.10: A comparison between the heading(a) and position(b) as measured by the drone, and the heading(c) and position(d) estimated by the Kalman filter. This data represents three flights from Drone 3.

Section 4.3. Examining the detailed view of the residual plot in Figure 4.12b shows that the $+X$ vanishing point was lost at about the 14 second mark, and the $+Y$ vanishing point was acquired just after the 16 second mark. The reason the residual is constant for short periods of time is due to the operation of the vanishing point algorithm. When the vanishing point algorithm fails to find a vanishing point, the Kalman filter algorithm only propagates the states and the state covariance matrix. There is no residual until the next measurement is taken and the states updated, so the value previously calculated residual is written to the log.

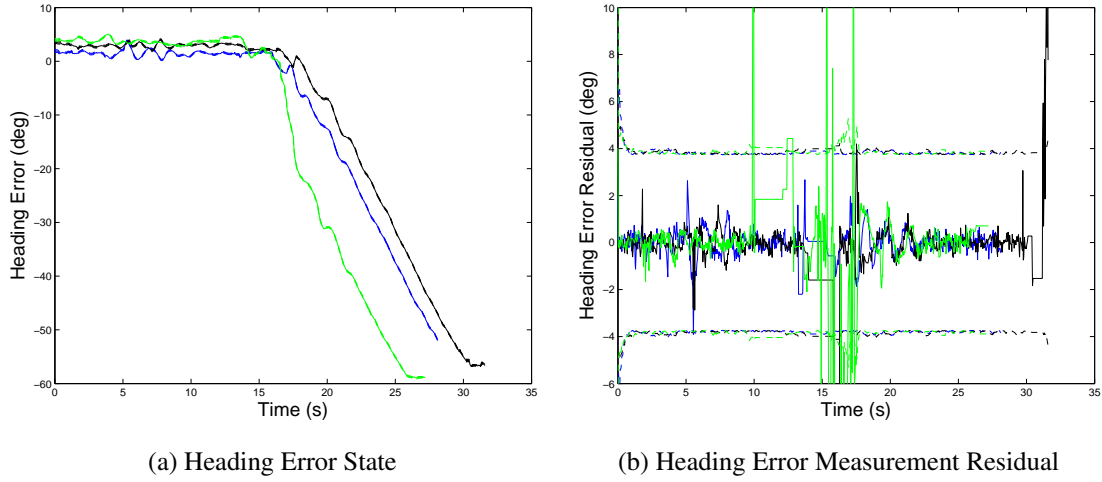(a) Heading Error State        (b) Heading Error Measurement Residual

Figure 4.11: An example of the heading error state from the vanishing point transition experiment. The heading error tracks the heading error as it grows continuously after the 90° turn(a). The heading error measurement residual shows that at about the 15 second point the drone is rotating between vanishing points, and that there are no valid measurements(b). The solid lines represent the heading error(a) and the measurement residual (b). The dashed lines represent $\pm 1\sigma$ in (a) and $\pm 1.5\sigma$ in (b).

#### 4.2.1.2 Heading Error Rate State.

The heading error rate state is shown in Figure 4.13. As expected the heading error rate remains centered around zero for the first leg of the flight, then jumps to match the heading error rate after the 90° turn. The large spikes that can be seen during the first leg of the flight correspond with the oscillations seen in the heading error state in Figure 4.11a. These are due to measurements based on false vanishing points that were near enough to the actual vanishing point not to be rejected. The high vanishing point measurement rate ensures that these points don't cause the system to fail. The heading error rate is only observable by the system during measurement updates, so the areas where the error rate is constant are where no measurements were available.

(a) Estimated Heading Error          (b) Heading Error Residual
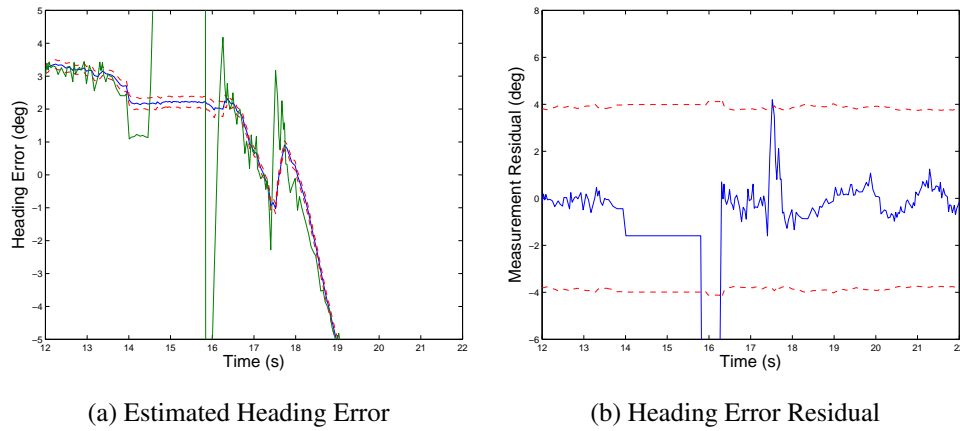
Figure 4.12: A detailed view of the heading error state and measurement residual during a typical vanishing point transition(a). In this case the vanishing point is lost near the 14 second point and not reacquired until just after the 16 second mark(b).
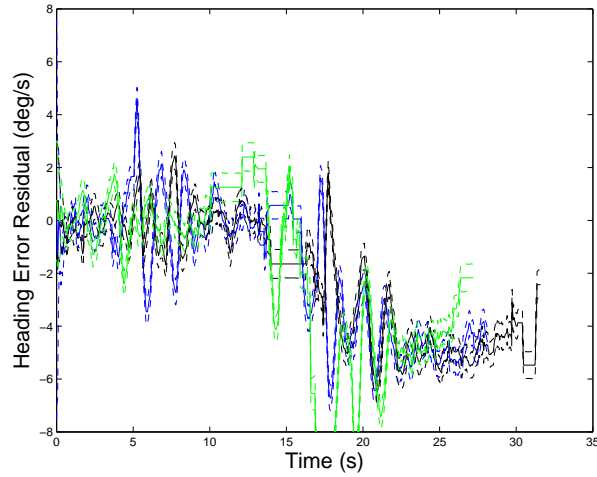


Figure 4.13: An example of the heading error rate state during the vanishing point transition experiment. The areas where the rate is constant are the periods where no measurements were available. The dashed lines represent $\pm 1\sigma$.

### *4.2.1.3 The Velocity Estimation States.*

The *X* velocity state is shown in Figure 4.14a and its residual in Figure 4.15a. Recall that the *X* and *Y* velocity states are estimates of the vehicle's velocity in the navigation frame. As expected, it shows the drone accelerating down the first leg of the course. When the drone makes its 90° turn the X velocity drops off near to zero as expected. The periods where the velocity is constant are the times where no vanishing point measurements were available. It is expected that the velocity state would propagate based on the last velocity measurement, however, due to the error mentioned in the introduction to this chapter, this didn't happen. The extent of this error and its effect are discussed further in Section 4.3.2. Examining the residual plot, shows that very few measurements were rejected, this is due to the overestimation of the velocity measurement and process noise pointed out in previous sections.

The *Y* velocity state and residual are shown in Figures 4.14b and 4.15b. As expected the *Y* velocity stays near zero during the first leg of the course, then shows an acceleration after the turn. The residual shows that unlike the *X* velocity measurements, the initial *Y* velocity measurements were rejected for a two second period starting at the 15 second mark. The system model developed for this drone in Chapter 3 did not account for user input. Here it can be seen that the change in direction was not anticipated by the model and treated as an error in velocity. After a period of about two seconds, the residual covariance grew enough that measurements were again incorporated into the model. The areas where the velocity is constant near the 15 second mark are due to the previously mentioned bug in the propagation portion of the Kalman filter algorithm.

## 4.3 Post-processing Analysis

A MATLAB program was created to investigate the effects of the errors in the real-time code and optimize the noise parameters for the process and measurement noise. This
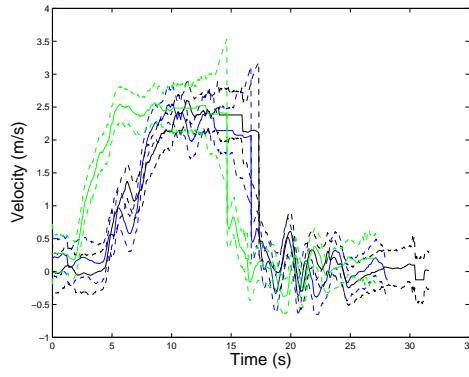
(a) X Velocity State
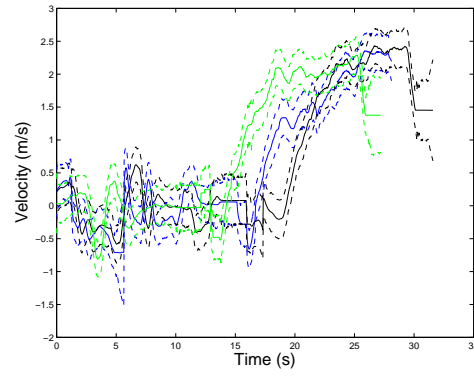
(b) Y Velocity State

Figure 4.14: An example of the *X*(a) and *Y*(b) velocity states from the vanishing point transition experiment. These plots represent data from three flights of Drone 3.
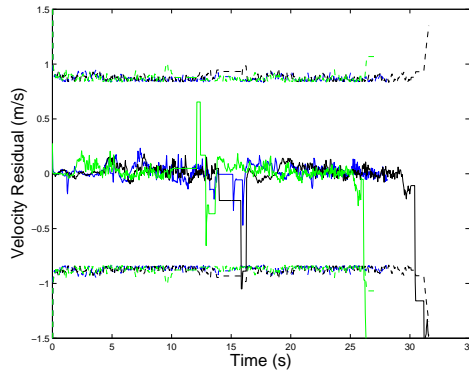


(a) X Velocity Measurement Residual
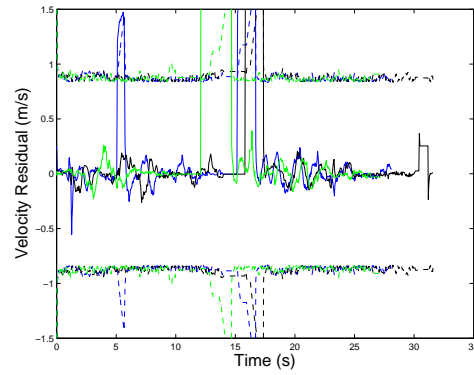
(b) Y Velocity Measurement Residual

Figure 4.15: The Measurement residuals for the velocity states. These plots represent data from three flights of Drone 3.

MATLAB program implements the Kalman filter algorithm and uses measurement data recorded by the real-time software to recreate the state estimates. Measurements from the

first flight of Ddrone 3 are used for the post-processing investigations in this section, so that the results will be comparable to the results presented in the previous section.

### 4.3.1 MATLAB Program Verification.

The first run was designed show that the post-processing results agree with the results from the online algorithm. A comparison of the real-time software and the MATLAB program are shown in Figure 4.16. As expected the plots lie nearly on top of each other. The numeric precision of the data logged by the real-time software was truncated when it was written to the log file. The slight differences between the MATLAB program and the real-time software are due to roundoff error.
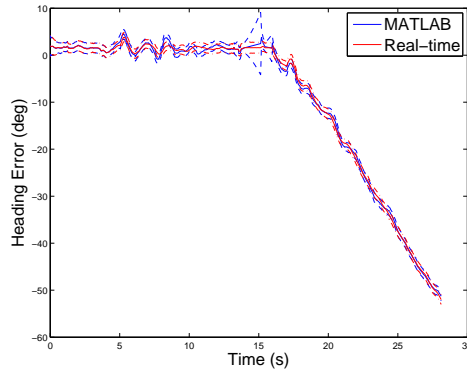
(a) Heading Error State            (b) Heading Error Residual
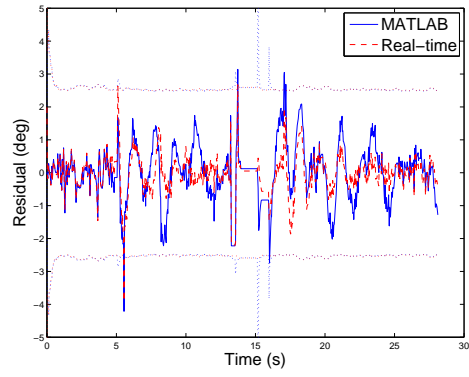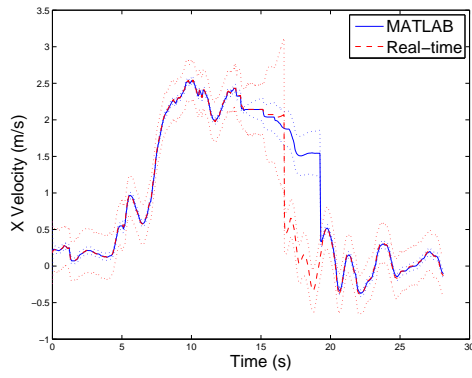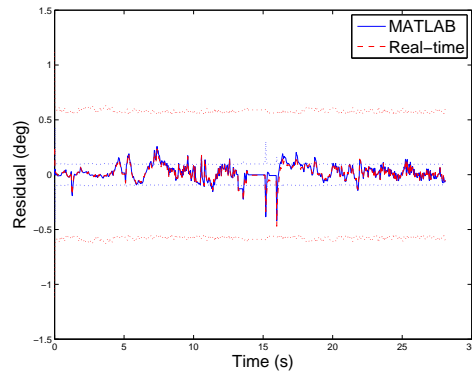
(c) X Velocity State            (d) X Velocity Residual
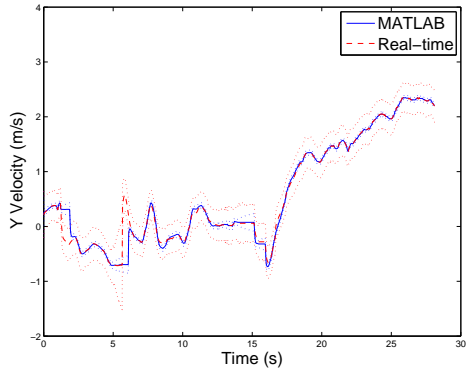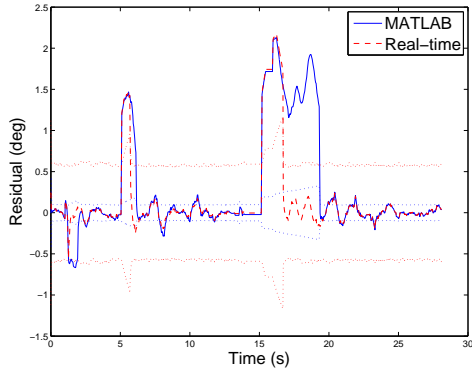
(e) Y Velocity State            (f) Y Velocity Residual

Figure 4.16: A comparison of the post-processing program and the real-time software. The solid blue line represents the MATLAB program results and the dashed red line represents the real-time software results. The dotted lines represent the $\pm 1\sigma$ bound.

### 4.3.2 *Effects of the Propagation Stage Error.*

In this section, the effect of error that was discovered in the propagation stage of the Kalman filter algorithm was investigated. The error was located in the real-time software's Kalman filter propagation code. When the state estimate, $\hat{\mathbf{x}}(t_i^-)$, and covariance matrix $\mathbf{P}(t_i^-)$ were propagated they were stored in temporary variables. These temporary variables were then passed to the update stage of the algorithm. The error occurred when the vanishing point algorithm failed to find a vanishing point and only the propagation stage was run. The values of the propagated states and covariance matrix in the temporary variables were lost. As a result the previous values of the states are copied to the next filter cycle rather that the propagated estimates. This error does not effect the filter when the vanishing point measurement is rejected for exceeding the residual threshold. The results of correcting the coding error are shown in Figure 4.17. It can be seen that when the bug is corrected the covariance begins to grow rapidly during the short period that no measurements are available. A detailed view of the heading error state is shown in Figure 4.18. Here it can be seen that the difference between the state estimates is minimal. Because the vanishing point algorithm is not dependent on the heading error state covariance, the overall effect on the heading estimation was minimal.

The effect of the error on the velocity states is due to the filter properly propagating the velocity, allowing the filter to better capture the deceleration trend leading up to the vanishing point transition. The *Y* velocity measurement rejections pointed out in Section 4.2.1.3, were reduced because the filter better predicted the change in velocity that occurred during the transition. The improvement to the velocity states also affected the estimated ground track as well. A comparison of the ground tracks is shown in Figure 4.19. It can be seen that the propagation bug accounts for about 1.5m of ground track error in this case.
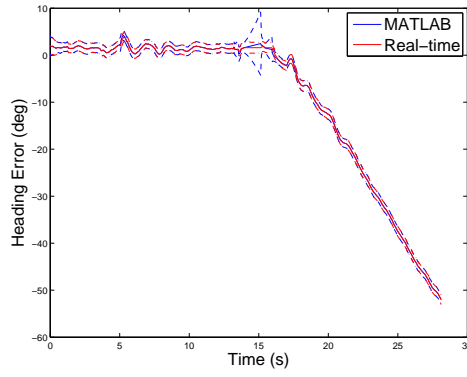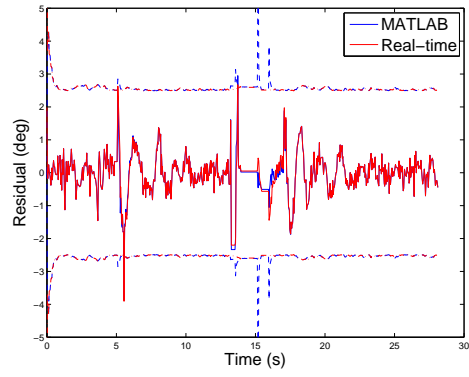
(a) Heading Error State      (b) Heading Error Residual
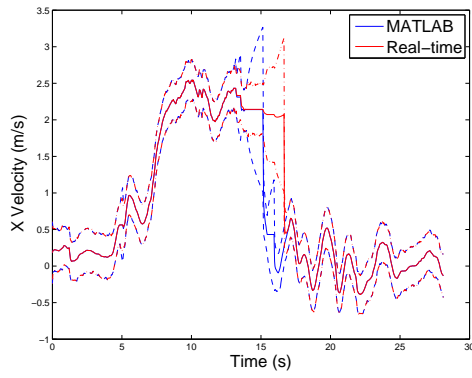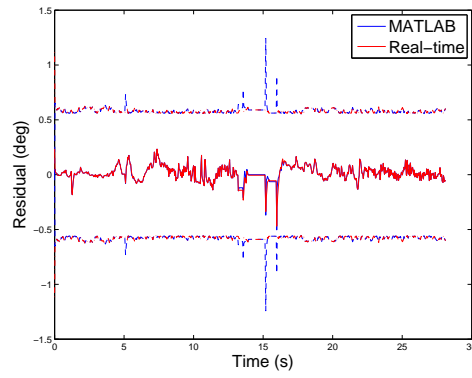
(c) X Velocity State      (d) X Velocity Residual
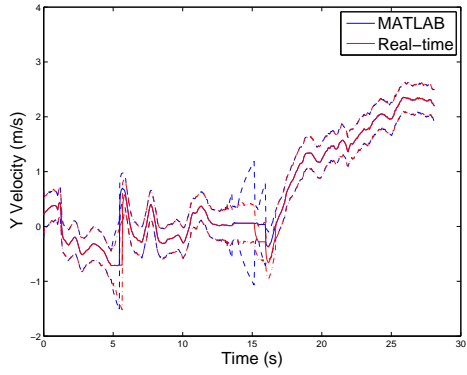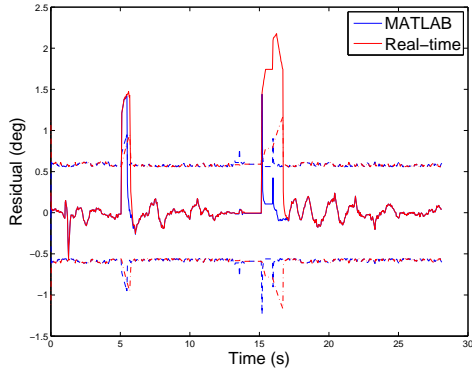
(e) Y Velocity State      (f) Y Velocity Residual

Figure 4.17: An example of the effect of the software error. The red lines represent the real-time software results, and the blue lines represent the corrected MATLAB program results
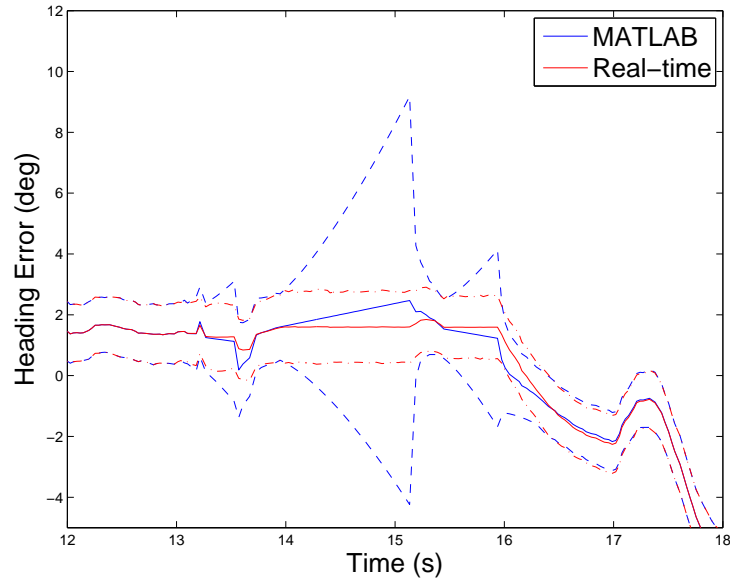
Figure 4.18: A detailed comparison between the heading error state from the real-time software and the MATLAB program. In the MATLAB program the propagation stage error has been corrected. The results of the real-time code are represented by the red lines and the results of the MATLAB program are represented by the blue lines.
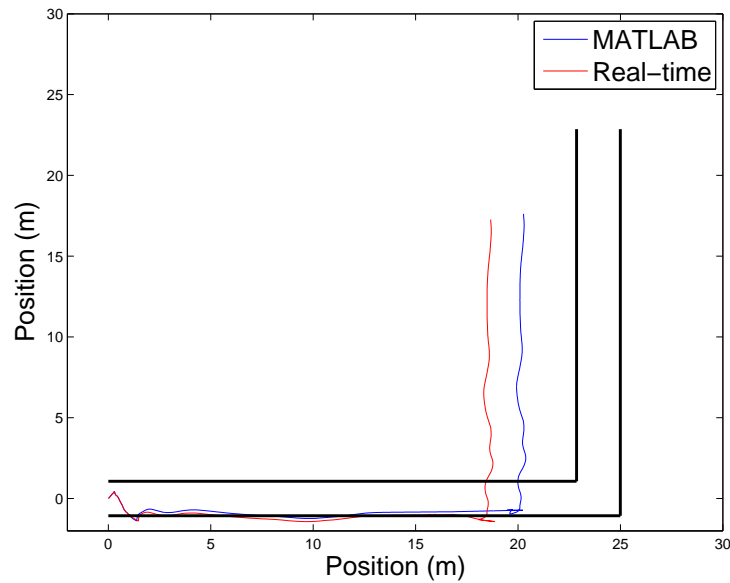


Figure 4.19: A comparison between the ground track estimated by the real-time software, in red, and the corrected MATLAB program in blue.

### 4.3.3 Kalman Filter Tuning.

This investigation is designed to show the effects of adjusting the process and measurement noise to bring the state and residual covariance closer to what is expected. The error correction from the previous section is used to generate these results.

The initial settings were chosen so that the real-time software would reliably transition between vanishing points regardless of which drone was being used. The wide variation in the drift rate of the drones' heading measurement forced the process noise strength for the heading error and heading error rate state to be relatively large. Using data from Drone 3's first flight, strengths of the process and measurement noise were adjusted to be

$$\mathbf{Q}(t) = \begin{bmatrix} 0.0001 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0.0625 & 0 \\ 0 & 0 & 0 & 0.0625 \end{bmatrix} \tag{4.1}$$

$$\mathbf{R}(t) = \begin{bmatrix} 0.000036 & 0 & 0 \\ 0 & 0.0064 & 0 \\ 0 & 0 & 0.0064 \end{bmatrix} \tag{4.2}$$

These values were chosen so that the square root of the residual covariance was approximately $1\sigma$. This required the residual rejection threshold be increased to $3\sigma$. For Drone 3's first flight, the system performs as expected. The heading error state, shown in Figure 4.20, closely follows the heading error state estimate from the real-time software. Furthermore, examining the measurement residual, shown in Figure 4.21, shows that the residual covariance is much closer to what is expected. Based on these results, it would appear that these estimates for $\mathbf{Q}(t)$ and $\mathbf{R}(t)$ are much more appropriate estimates for this system. However, when the rest of the flight test data for Drone 3 is reprocessed it becomes evident that these settings are not ideal for the rest of the drones.

(a) Heading Erro State

(b) Heading Erro State Detail
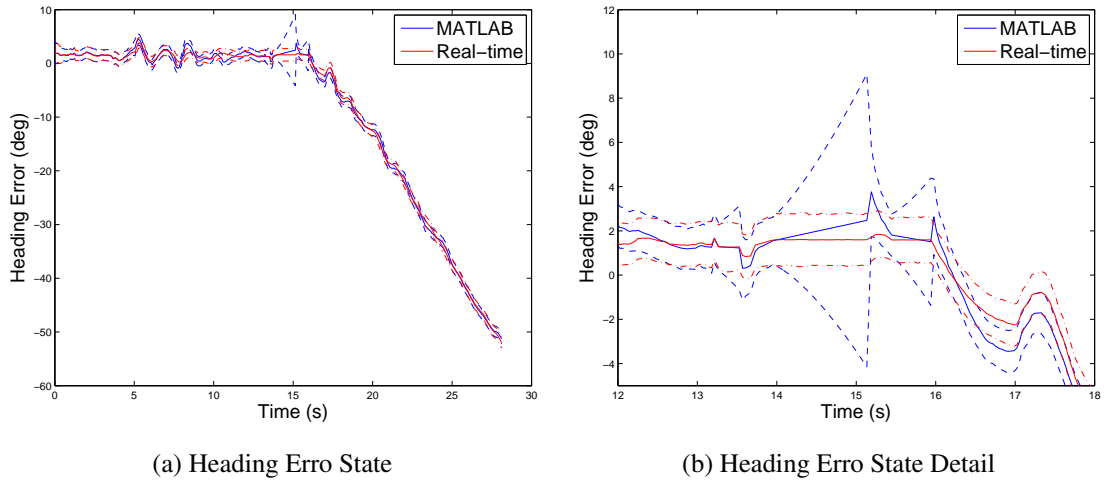
Figure 4.20: The heading error state for Drone 3's first flight where the process and measurement noise strength have been reduced. The results from MATLAB are shown in blue, and the real-time software results are shown in red.
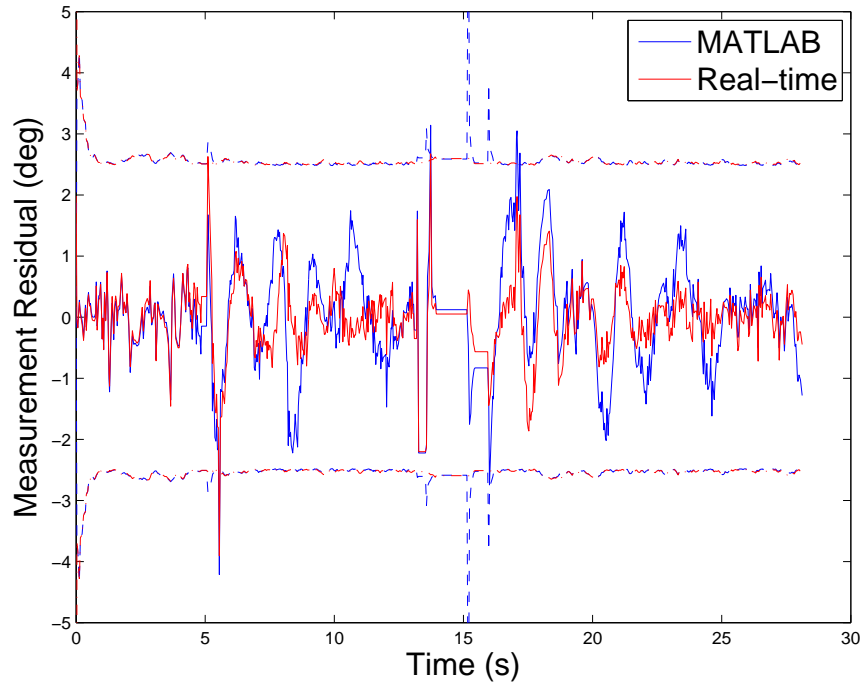


Figure 4.21: The heading error measurement residual for Drone 3's first flight where the process and measurement noise strengths have been reduced. The results from MATLAB are shown in blue, and the real-time software results are shown in red.

The data from Drone 2's flight tests was reprocessed to examine if these settings could be applied to the other drones. The measurement residuals of the reprocessed data for all of the flight tests for Drones 2 and 3 are shown in Figure 4.22. Examining these plots reveals that for one of Drone 3's flights the residual failed to converge after the turn. This indicates that the system would have failed to track a vanishing point. In the case of Drone 2, the residual converged after the turn for each flight, but in the case of its third flight this did not occur until the vehicle had traveled almost the entire second leg. Recall from Chapter 3 that the vanishing point algorithm ignores lines that are not with in 5° of the predicted vanishing point's location. A detailed view of Drone 2's heading error state during the period of time between 12 and 24 seconds of its third flight is shown in Figure 4.22. Here it can be seen that the difference between the heading error state estimates is greater than 5°, which indicates that the vanishing point algorithm would not have been able to require the vanishing point.



(a) Drone 2          (b) Drone 3
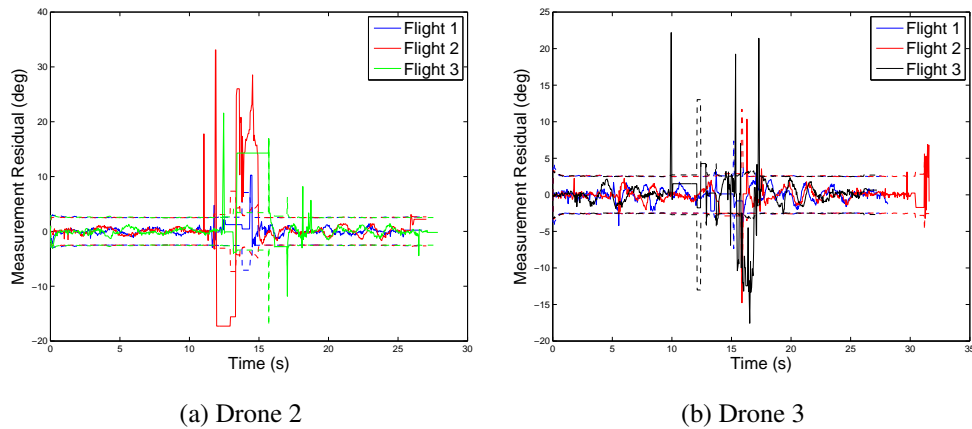
Figure 4.22: The heading error state measurement residuals from the reprocessed flight data for each flight of Drones 2(a) and 3(b). The blue lines represent the data from flight one, the red line represents flight two, and the green line represents the third flight.
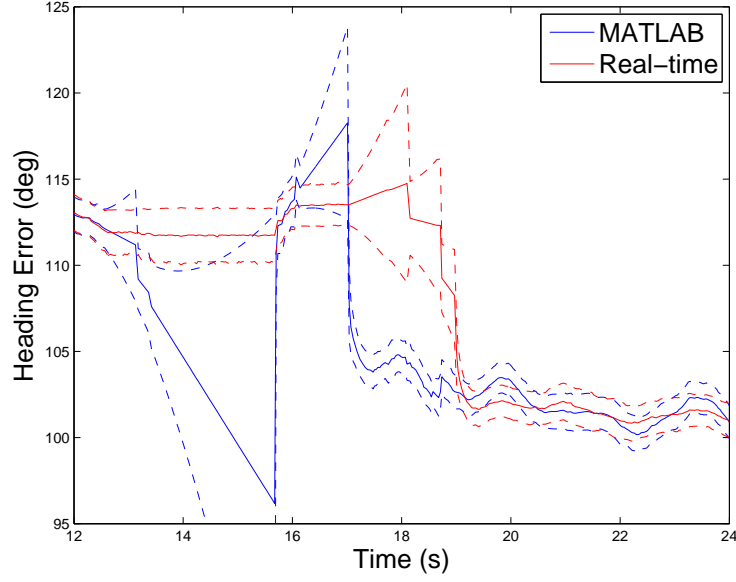
Figure 4.23: A detailed view of the heading error state for Drone 2's third flight after being reprocessed. The results from the real-time software are shown in red and the results from MATLAB are shown in blue.

For the velocity state tuning, the heading values for the process and measurement noise for the heading error state and heading error rate state were restored to their previous values. The strength of the process and measurement noise for the velocity states were adjusted in the same manner as the heading error states. The the final estimates for process and measurement noise strength are

$$\mathbf{Q}(t) = \begin{bmatrix} 0.001 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.0196 & 0 \\ 0 & 0 & 0 & 0.0196 \end{bmatrix} \tag{4.3}$$

$$\mathbf{R}(t) = \begin{bmatrix} 0.0016 & 0 & 0 \\ 0 & 0.0049 & 0 \\ 0 & 0 & 0.0049 \end{bmatrix} \tag{4.4}$$

65

The results of these adjustments are shown in Figure 4.24. The measurement residual plots show that the measurement residual covariance is closer to what is expected. These settings required that the rejection threshold on the velocity states be increased to $4\sigma$. The effect of the tuning is more easily seen in the drone's estimated ground track shown in Figure 4.25. Here it can be seen that accurately tuning the velocity noise terms results in improved position estimation. When these settings were used to reprocess the data from Drone 2, an improvement in its position estimate were also seen.



(a) X Velocity Residual                    (b) Y Velocity Residual
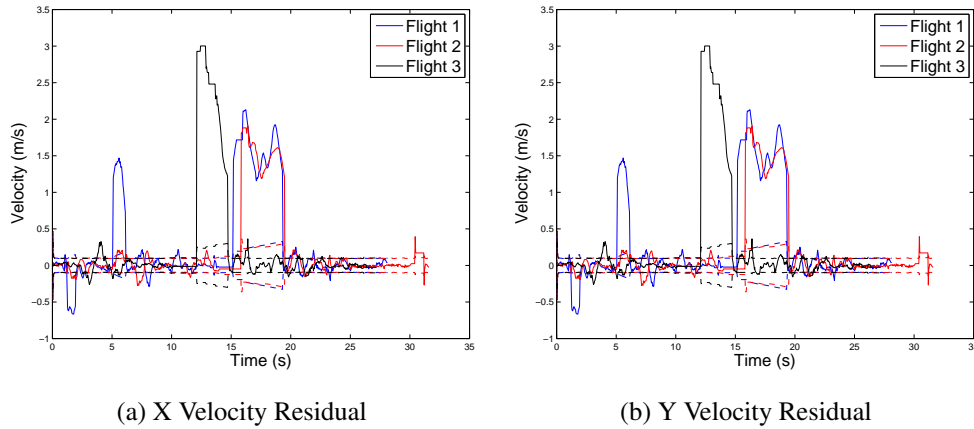
Figure 4.24: The results of adjusting the process and measurement noise estimates for the velocity states. The solid lines represent the measurement residual, the dashed lines represent the square root of the residual covariance.

## 4.4   Summary

This chapter presented three experiments designed to validate the navigation software design and characterize its performance in multiple environments. The real-time software worked as expected and provides a reliable heading estimate for the vehicle. During the evaluation of the test data, an error was discovered in the Kalman filter propagation
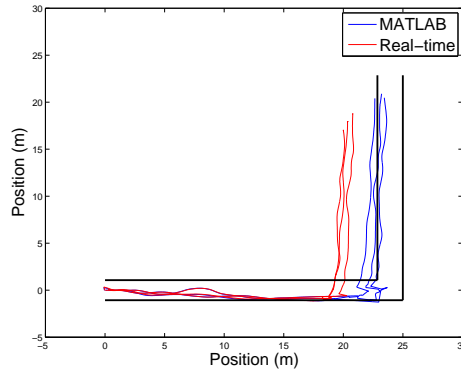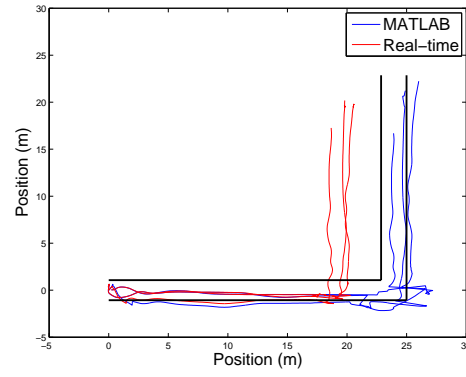
(a) Drone 2                    (b) Drone 3

Figure 4.25: The results of reprocessing the data from Drone 2(a) and Drone 3(b) with the adjusted noise parameters. The red lines represent the ground track estimated by the real-time software, and the blue lines represent the ground track estimated by MATLAB.

stage. The error did not cause the system to fail. Because of the high measurement rate, the amount of time the filter was left to propagate was minimal. The post processing investigations showed while the noise parameters for the heading error and heading error rate states were not mathematically ideal, they were required to ensure the reliable operation of the system. The post-processing investigation did show that with improved tuning parameters on the velocity states, the system was able to better estimate its position in the navigation frame.

67

## 5 Conclusion

THE goal of this research was to develop an algorithm that could operate in real time to control the heading of an ARDrone, and estimate its position and velocity. In Chapter 3 a navigation algorithm was developed to accomplish this goal. This involved adapting Prahl's vanishing point algorithm to run in real-time and provide an estimate of the drone's heading. In addition to developing the vanishing point algorithm, a system model for the ARDrone was developed. This system model was used to design an extended Kalman filter to estimate the drone's heading, position and velocity in a local level navigation frame. The final component developed in chapter three was a proportional controller that used the heading estimate provided by the EKF to keep the drone on a constant course.

In Chapter 4, three experiments were conducted. The first two experiments were flight tests conducted in the hallways of a government building. The constant heading flight tests demonstrated that the navigation algorithm developed in Chapter 3 was capable of holding the drone on a constant course. The vanishing point transition flight test demonstrated that the vehicle could travel down a hallway, turn $90°$ start tracking a new vanishing point, and continue navigation. The post-processing investigation demonstrated that the noise model developed in Chapter 3 for the heading error and error rate states could not be improved without degrading the performance of the system. This investigation did show that an improved noise model for the velocity states improved the algorithms ability to estimate the drone's position and velocity.

During the experiments in Chapter 4, an error in the Kalman filter propagation code was discovered. This error prevented the states and covariance matrix from propagating correctly. The overall impact of this error was shown to be minimal over the experiments

conducted here. The high measurement rate prevented the filter from being left to propagate for long periods of time.

## 5.1 Future Work

The experiments in Chapter 4 demonstrated that there are several improvements to the system that should be investigated in future work. In addition to improving the system model, there is a need to incorporate another set of velocity measurements into the model.

### 5.1.1 *Improved System Model.*

The system model developed for this research used fixed quantities for the white noise strength in the noise model. The post-processing investigation showed that because the quantities were fixed, they had to be much larger in order to account for the uncertainty caused by user input. Developing a dynamic noise model, where the noise parameters of the heading error and heading error rate states were functions of the user commanded yaw rate, would allow for the system to react to the uncertainty caused by user input during turns.

### 5.1.2 *Improved Velocity Estimation.*

In order to reduce the uncertainty in the vehicle's position, an additional means of estimating the vehicle's velocity needs to be incorporated. This would also allow the vehicle to operate over a wider range of floor surfaces. Methods such as feature tracking or "optical sonar" could be included to provide a more robust estimation of velocity and improve localization [13].

### 5.1.3 *Improved Vanishing Point Algorithm.*

The vanishing point algorithm used in this research did not take into account the uncertainty of the heading estimate used to predict the location of the vanishing point. Instead of using a fixed search area, the vanishing point algorithm should determine the size of the search space based on the heading error state covariance. This would improve the vehicle's ability to reacquire the vanishing point after a rapid turn.

## 5.2 Final Conclusion

This research provided a real-time implementation of a vision-aided navigation algorithm that has the potential to be imbedded in a small aerial vehicle. A vehicle equipped with this technology in the future will provide a low cost solution to mapping unknown man-made structures. Additionally, this research showed that the ARDrone is a capable low cost platform for vision-aided navigation research.

# Bibliography

[1] "Ascending Technologies". URL http://www.asctec.de.

[2] "Mikrokopter", October 2012. URL http://www.mikrokopter.com/ucwiki/en/Software.

[3] Bills, Cooper, Joyce Chen, and Ashutosh Saxena. "Autonomous MAV Flight in Indoor Envronments using Single Image Perspective Cues". *Robotics and Automation (IRCA), 2011 IEEE International Confrence on*, 5776–5783. 2011.

[4] Bonin-Font, Francisco, Alberto Ortiz, and Gabriel Oliver. "Visual Navigation for Mobile Robots: A Survey". Online, May 2008.

[5] Bouguet, Jean-Yves. "MATLAB Camera Calibration Toolbox", July 2010. URL http://www.vision.caltech.edu/bouguetj/calib_doc/.

[6] Choi, Young-Ho and Se young Oh. "Visual Sonar Based Localization using Particle Attraction and Scattering". *Mechatronics and Automation, 2005 IEEE International Conference on*. 2005.

[7] Coremen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[8] DeSouza, Guilherme N. and Kak Avinash C. "Vision for Mobile Robot Navigation". *IEEE Transactions on Patteren Analysis and Machine Intelligence*, 24(2):237–267, February 2002.

[9] Forsyth, David A. and Jean Ponce. *Computer Vision A Modern Approach*. Prentice Hall, 2003.

[10] Hartley, Richard and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[11] Hrabar, S., G.S. Sukhatme, P. Corke, K. Usher, and J. Roberts. "Combined Optic-flow and Stereo-based Navigation of Urban Canyons for a UAV". *Intelligent Robots and Systems, 2005 IEEE/RSJ International Conference on*. 2005.

[12] Kleinberg, Jon and Éva Tardos. *Algorithm Design*. Pearson Education, Inc, 2006.

[13] Lenser, Scott and Manuela Veloso. "Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision". *Intl. Confrence on Intellegent Robots and Systems*. IEEE/RSJ, 2003.

[14] Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, volume 1. Navtech Book and Software Store, 1994.

[15] Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, volume 2. Navtech Book and Software Store, 1994.

[16] Netter, T. and N. Franceschini. "A Robotic Aircraft that Follows Terrain Using a Neuromorphic Eye". *Intelligent Robots and Systems, IEEE International Confrence On*. 2002.

[17] Piskorski, Stephane, Nicolas Brulez, Pierre Eline, and Frederic D'Hayer. "The ARDrone Developer Guide". May 2012.

[18] Prahl, Dayvid. *Coupling Vanishing Point Tracking with Inertial Navigation to Estimate Attitude in a Structured Environment*. Master's thesis, Air Force Institute of Technology, 2011.

[19] Se, S., D.G. Lowe, and J. Little. "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks". *International Journal of Robotics Research*, 2002.

[20] Shapiro, Linda G. and George C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[21] Sim, R. and G. Dudek. "Learning Generative Models of Scene Features". *Computer Vision and Pattern Recognition, Proceedings of the 2001 IEEE Computer Society Conference on*. 2001.

[22] Titterton, D. H. and J. L. Weston. *Strapdown Inertial Navigation Technology*, volume 207. American Institute of Aeronautics and Astronautics, 2nd edition, 2009.

[23] Tomono, M. "3-D Map Building Using Dense Object Models with SIFT-based Recognition Features". *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. 2006.

[24] Zhang, Zhengyou. "A Flexible New Technique for Camera Calibration". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330 – 1334, November 2000.

| REPORT DOCUMENTATION PAGE | | | | *Form Approved* *OMB No. 0704–0188* |
|---|---|---|---|---|

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

| 1. REPORT DATE (DD–MM–YYYY) 21-03-2013 | 2. REPORT TYPE Master's Thesis | 3. DATES COVERED *(From — To)* Oct 2011 – Mar 2013 |
|---|---|---|

| 4. TITLE AND SUBTITLE Real-time Heading Estimation using Perspective Features | 5a. CONTRACT NUMBER |
|---|---|
| | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |

| 6. AUTHOR(S) Dean, James W., Captain, USAF | 5d. PROJECT NUMBER |
|---|---|
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |

| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/ENY) 2950 Hobson Way WPAFB OH 45433-7765 | 8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENG-13-M-13 |
|---|---|

| 9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank | 10. SPONSOR/MONITOR'S ACRONYM(S) |
|---|---|
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES     This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

There are a large number of commercially available quad-rotor helicopters available from various manufacturers. All of these systems rely on a low cost MEMS based inertial measurement system for stabilization and navigation. These low cost inertial systems are all subject to rapid error growth in their attitude and position estimates unless bounded by external measurements. This thesis created real-time algorithm to integrate measurements from visual cues with measurements from onboard sensors to estimate the attitude position and velocity of a quad-rotor helicopter in a local navigation frame, a system model for the ARDrone, and a feed-back controller for the vehicle's heading. The ARDrone, by Parrot SA, is a low cost quad-rotor helicopter that comes equipped with a variety of sensors including a forward-looking high-definition camera. The vehicle is capable of using its onboard sensors to adequately constrain the errors for pitch and roll in all environments, however the yaw axis is still subject to drift. This work utilizes a RANSAC based vanishing point detection algorithm to provide a reliable heading reference and integrates the vanishing point based heading measurements with the system's on-board heading measurements through an extended Kalman filter. In addition to estimating the drone's heading, the Kalman filter also estimates the position and velocity of the drone as it moves through its environment. This system was able to provide a heading reference with an error of one degree for the drone and was shown to be capable of transitioning between vanishing points when the vehicle needed to change direction. The system also demonstrated that it was capable of generating an estimate of position and velocity. However because the position error was on the order of one meter, the estimate was not accurate enough for autonomous navigation.

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON Dr. John F. Raquet |
|---|---|---|---|---|---|
| a. REPORT U | b. ABSTRACT U | c. THIS PAGE U | UU | 90 | 19b. TELEPHONE NUMBER (Include Area Code) (937)255-3636, ext 4580 |

Standard Form 298 (Rev. 8–98)
Prescribed by ANSI Std. Z39.18